Agile and conventional methodologies: An empirical investigation of their impact on software quality parameters

by

DONALD MBUYA PENN

Submitted in accordance with the requirements for

the degree of

MASTER OF SCIENCE

in the subject

COMPUTING

at the

University of South Africa

Supervisor: Prof. Ernest Mnkandla

January 2016

ABSTRACT

The advent of agile methodologies has brought about an illuminating debate in Software Engineering, particularly with regard to software quality. Some studies have reported that agile methodologies do improve software quality when compared to traditional methodologies; other studies have been inconclusive or contradictory, while others have argued that empirical evidence is limited. This study sought to investigate the correlation between agile methodologies when compared to traditional methodologies for selected software quality parameters. The research design was causal comparative, as well as correlational. The approach was quantitative, using a survey as the data collection method. SPSS was used to conduct descriptive and correlational analysis for 106 responses received.

The main findings were that there was a statistically significant relation between traditional methodology use and ease of system testing (p=0.014); a statistically significant relation between traditional methodology use and timeliness (p=0.02); a statistically significant relation between software quality standards used and ease of system testing (p=0.017); a statistically significant relation between active stakeholder participation on projects and ease of system interactivity (p=0.047); and a statistically significant relation between mandatory workshop attendance or training and ease of system navigation (p=0.031).Claims that agile methodology use leads to improved software quality for selected quality parameters could not be empirically validated. The association between most of the selected software quality criteria in relation to methodology use in general was not apparent. Agile methodologies are suitable in small environments. Scrum was the most widely used agile methodology by far. The popularity and adoption state of XP showed a significantly decreasing trend. Traditional and agile methodologies combined are being used (47%) more than any other methodology. Agile methodology use (28%) surpassed traditional methodology use (19%). A suitable consensus definition for agile methodologies did not emerge from the data collected. The most suitable project life cycle model was evolutionary, incremental and iterative. ‘Other’ methodologies, meaning customised agile or SDLC, are suitable, as the environment becomes increasingly large and complex. Only 13% of organisations surveyed have an agile experience of six years and beyond. Based on these findings and gaps in the literature, implications and recommendations for further research areas are proposed, where the findings and contributions of this study are found to be relevant to practice for application and to academia for further research.

Key terms

Agile, conventional, traditional, methodologies, empirical, investigation, impact, software quality, relationship, correlation, significant, compared, association, criteria and parameter.

DECLARATION

Student number: 35552808

I declare that **Agile and conventional methodologies: An empirical investigation of their impact on software quality parameters** is my own original work, and that all the sources that i have used or quoted have been indicated and acknowledged by means of complete references.

08/01/2016

_____ _____

SIGNATURE DATE

(Mr Donald Mbuya Penn.)

ETHICS CLEARANCE

UNISA | college of science, engineering and technology

Dear Mr. Donald Mbuya Penn (35552608)

Date: 2015-04-15

Application number:
044/DMP/2015

REQUEST FOR ETHICAL CLEARANCE: (Agile and conventional methodologies: An empirical investigation of their impact on software quality parameters)

The College of Science, Engineering and Technology's (CSET) Research and Ethics Committee has considered the relevant parts of the studies relating to the abovementioned research project and research methodology and is pleased to inform you that ethical clearance is granted for your research study as set out in your proposal and application for ethical clearance.

Therefore, involved parties may also consider ethics approval as granted. However, the permission granted must not be misconstrued as constituting an instruction from the CSET Executive or the CSET CRIC that sampled interviewees (if applicable) are compelled to take part in the research project. All interviewees retain their individual right to decide whether to participate or not.

We trust that the research will be undertaken in a manner that is respectful of the rights and integrity of those who volunteer to participate, as stipulated in the UNISA Research Ethics policy. The policy can be found at the following URL:
http://cm.unisa.ac.za/contents/departments/res_policies/docs/ResearchEthicsPolicy_apprvCounc_21Sept07.pdf

Please note that the ethical clearance is granted for the duration of this project and if you subsequently do a follow-up study that requires the use of a different research instrument, you will have to submit an addendum to this application, explaining the purpose of the follow-up study and attach the new instrument along with a comprehensive information document and consent form.

Yours sincerely

Prof Ernest Mnkandla
Chair: College of Science, Engineering and Technology Ethics Sub-Committee

Prof IDG Moche
Executive Dean: College of Science, Engineering and Technology

RECEIVED
2015 -04- 15
OFFICE OF THE EXECUTIVE DEAN
College of Science, Engineering and Technology

University of South Africa
College of Science, Engineering and Technology
The Science Campus
C/o Christiaan de Wet Road and Pioneer Avenue,
Florida Park, Roodepoort
Private Bag X6, Florida, 1710
www.unisa.ac.za/cset

UNISA | college of science, engineering and technology

ACKNOWLEDGEMENTS

All praise and glory is due to **Almighty God**, the bestowal of knowledge and wisdom, for making it possible for me to complete this dissertation.

My sincere and heartfelt gratitude goes to the following:

Professor Ernest Mnkandla my supervisor, for your expert supervision of this dissertation and valuable guidance throughout this study.

Dr Joke van Niekerk, for the valuable role you played in supervising the research proposal.

Jacob Mays, for all your valuable assistance with the statistical analysis.

Genevieve Wood, for your meticulous and prompt language editing of this dissertation.

Dr Monica Otu, for proofreading this dissertation.

My wife **Jessica**, for your support and understanding.

My daughter **Princess Pearl Mengwi,** for your love and patience.

My **family and friends**, for your love and encouragement.

To all **participants** who took part in the pilot survey, for your valuable input.

To all **respondents** of the survey for providing valuable data.

DEDICATION

This dissertation represents a significant milestone in my life and professional career.

I dedicate it to the following:

To my late father, **Mbuya W Jeremiah** from whom i drew inspiration.

To my later brother, elder **Mbuya Emmanuel Teneng** for the indispensable role you played in shaping my vision.

To my late sister, elder **Mbuya Florence Tewah,** for your love and care.

To my late father in law, **Agwe Peter Tamba,** for your words of wisdom.

To my lovely mother, **Mbuya Regina,** for your good genes.

To my lovely step mother, **Mbuya Mary,** for your love and care.

To my darling wife, **Jessica,** for your enduring support and encouragement.

To my lovely daughter, **Princess Pearl Mengwi**.

Table of Contents

List of Tables

List of Figures

CHAPTER 1: BACKGROUND AND INTRODUCTION

1.1 Background

Information systems are conventionally developed using a life cycle called the systems development life cycle (SDLC), or the Waterfall Model. An information system is an array of people, data, processes and information technology that interact to collect, store and disseminate the information required to support an organisation or individual (Whitten and Bentley,2007).

The approach just mentioned emerged in the 1970s, and since then, many variants have come into existence (Avison and Fitzgerald, 2006; Schach, 2010; Zhang, Hu, Dai and Li, 2010). In its classic form, the SDLC is made up of the following sequential phases: feasibility study; systems investigation; systems analysis; systems design; implementation; review; and maintenance (Avison and Fitzgerald, 2006). In other words, the aforesaid phases constitute what is referred to as a system development methodology. It was the traditional or conventional way to engineer information systems prior to the advent of more contemporary methodologies. A system development methodology is a disciplined and systematic process that is followed in developing an information system (Pressman, 2010). It consists of methods, techniques and tools, as well as an underlying philosophy (Avison and Fitzgerald, 2006). For the purpose of this study, system development methodology and methods are used interchangeably and mean the same thing.

The SDLC is a tried and tested methodology that has stood the test of time (Avison and Fitzgerald, 2006). It is more manageable than other methodologies, and enforces discipline in the development process with the use of checkpoints. However, the greatest weakness of the Waterfall Model is its inflexibility and failure to respond to change (Huo, Verner, Zhu and Barbar, 2004; Avison and Fitzgerald, 2006; Sommerville, 2007; Barlow et al., 2011; Kendall, Sue and Kendall, 2010; Leau, Loo, Tham and Tan, 2012). As a complement to the SDLC or Waterfall model, the Rational Unified Process (RUP), the Spiral Model, the V-Model and other methodologies have since emerged to address some, but not all, the weaknesses of the Waterfall Model (Schach, 2010).

All the foregoing methodologies are called traditional or conventional software development methodologies, and are classified into the category of heavyweight methodologies, due to the fact that they make use of enormous amounts of documentation (Nikiforova, Nikulsins and Sukovskis, 2009).

In 2001, a new and somewhat controversial paradigm to software development, called agile, emerged (Cunningham, 2001; Huo, Verner, Zhu and Barbar, 2004; Schach, 2010). This

approach emerged as a response to the weaknesses and limitations of conventional methods (Avison and Fitzgerald, 2006; Abrahamsson, Conboy and Wang, 2009; Rao, Naidu and Chakka, 2011; Barlow et al. 2011; Kendall, Sue and Kendall, 2010).

A group of experienced software engineering practitioners came together in 2001 and created the Agile Manifesto. The manifesto stated that agile development should focus on the following core values (Cunningham, 2001):

- *Individuals and interactions*, over *processes and tools*;
- *Working software*, over *comprehensive documentation*;
- *Customer collaboration*, over *contract negotiation*; and
- *Responding to change*, over *following a plan*.

They argued that even though practices they suggest deemphasising are also important, they are not as highly valued *(see italicised words in bulleted paragraph above*). Based on this manifesto, they proceeded to formulate the following twelve principles that define criteria for the agile software development process (Cunningham, 2001).

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity. The art of maximising the amount of work not done is essential.
11. The best architectures, requirements, and designs emerge from self-organising teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Since the inception, articulation and promulgation of the Agile Manifesto in 2001, different researchers have, over the years, distinguished between agile and traditional methods using different perspectives. These differences are summarised in table 1.

	Perspective	Traditional	Agile	Reference
1.	Development life cycle.	Tends to be fairly linear.	Iterative and evolutionary.	Charvat, 2003; Nerur, Mahapatra, and Mangalaraj, 2005.
2.	Business Requirements.	Fairly stable, clear, concise and documented.	Emergent, discovered progressively.	Boehm and Turner, 2004.
3.	Software Architecture.	Designed for current and the future.	Design for what is presently essential.	Boehm, 2002; Wysocki, 2011.
4.	Management approach.	Process centric, command and control.	People centric, leadership and collaboration.	Boehm and Turner, 2005; Vinekar, Slinkman and Nerur, 2006.
5.	Extent of documentation.	Enormous and heavily dependent on explicit knowledge.	Light documentation and mostly dependent on tacit knowledge.	Boehm and Turner, 2005.
6.	Goal.	Predictability and optimisation.	Exploration and adaptation.	Dyba and Dingsoyr, 2009.
7.	Change.	Tend to resist change.	Embrace change.	Boehm and Turner, 2003.
8.	Team members.	Distributed teams of specialists; Plan-oriented, adequate skills access to external knowledge.	Agile, knowledgeable, collocated and collaborative; Co-location of generalist senior technical staff.	Boehm, 2002; Sherehiy, Karwowski and Layer, 2007.
9.	Team organisation.	Teams are pre-structured.	Self-organising teams.	Leffingwell, 2007.
10.	Client involvement.	Client involvement is mostly low and passive.	Client is mostly onsite, active and proactive.	Highsmith and Cockburn, 2001.

11.	Organisational culture.	Culture is mostly command and control.	Leadership and collaborative.	Highsmith, 2002; Nerur, Mahapatra and Mangalaraj, 2005.
12.	Development process.	Anticipatory, predictability and high assurance.	Adaptive and flexible	Salo, and Abrahamsson, 2007; Leffingwell, 2007.

Table 1 : Traditional and agile perspectives on software development.

As table 1 depicts, agile is a paradigm that deviates from the conventional, plan based approaches to software development (Highsmith, 2002; Dyba and Dingsoyr 2008a; Bhasin, 2012). Agile methods contrast with traditional methods, which are dependent on a set of pre-planned processes and on-going documentation, written progressively to guide further development (Sommerville, 2007; Nikiforova et al., 2009; Rao, Naidu and Chakka, 2011). They therefore represent a major transition from conventional plan-based methodologies to more customer-centric approaches (Boehm and Turner, 2005; Vinekar, Slinkman and Nerur, 2006; Bhasin, 2012). They emerged as a response to formal, heavy, less efficient and bureaucratic software development practices (Boehm and Turner, 2005; Kendall, Sue and Kendall, 2010).

1.2 Problem statement

Like any new methodology, agile proponents assert that agile methodologies are a superior paradigm used to develop systems when compared with their predecessors. Among some of the strengths forwarded are: improved software quality (Santos, Bermejo, Oliveira and Tonelli, 2011; Rao, Naidu, and Chakka, 2011); focus on customer needs; adaptability to changing requirements; faster development times (Barlow et al., 2011); and flexibility (Austin and Devin, 2009).

However, despite the reported benefits, several studies have found that there are many problems and difficulties encountered with agile adoption, as well as noting the difficulties of putting the proposed benefits into practice (Barlow et al., 2011; Leau, Loo, Tham and Tan, 2012; Prause and Durdik, 2012; Asnawi, Gravell, and Wills, 2012; Pathak and Saha, 2013; Twidale and Nichols, 2013). This pre-existent deadlock then raises questions as to whether the claimed benefits of agile are purely theoretical, or also pragmatic. Some organisations adopt them to gain a competitive advantage, while others are still sceptical as to whether agile development is beneficial (Barlow et al., 2011) or whether the benefits of agile outweigh the cost (Ambler, 2008; Selic, 2009).

In addition, the benefits of agile methods and their impact have not been overwhelmingly convincing. Some studies have reported that they do improve product quality, other studies have

been inconclusive, and yet others have been contradictory. Abrahamsson, Oza and Siponen (2010) argue that empirical evidence after ten years of application remains quite limited. Rodríguez, Yague, Alarcon and Garbajosa (2009) noted that several studies have described success stories using agile methods but that these studies do not provide enough contextual information, or function merely as a type of 'lessons learnt' report. Based on study design, study quality, consistency and directness, Dyba and Dingsoyr (2008b) uncovered that the strength of evidence on the benefits and limitations of agile methods is very low.

Hummel (2014) conducted a systematic literature review in the field of agile information system development, encompassing 482 papers. Building and extending on previous reviews, he found that most of the research methods employed where qualitative, and recommended that more confirmatory quantitative studies are needed to corroborate previous or initial findings. He also uncovered that the theoretical underpinnings of agile methodologies are lacking.

The forgone arguments clearly illustrate that there is not yet consensus regarding the claimed benefits of agile development. Much has been said and written about the agile development process, but there is a dearth of literature when it comes to what happens after agile development Adoption (Abrahamsson, Conboy and Wang, 2009).

Several studies have been published on the adoption of agile methods, but research particularly in the area of agile quality assurance has been lagging behind (Bhasin, 2012).

There is therefore a need for a better understanding of agile methods beyond the adoption stage (Abrahamsson, Conboy and Wang, 2009). The preceding commentary is compelling enough to warrant an empirical investigation of agile methods, post adoption, because their merits are not yet clear, and shades of grey still exist.

An empirical study is a test that compares what we believe or claim, to what we observe (Perry, Porter and Votta, 2000). Empirical research seeks to study phenomenon and draw meaningful conclusions in a way that predictions can be made (Sjoberg, Dyba and Jorgensen, 2007). Therefore, the purpose of this study is to conduct a practical investigation of the impact of agile when compared to traditional methods on selected software quality parameters.

To be more elaborate, the study aims to investigate their impact on some system and information quality criteria in organisations that have used conventional methodologies in the past, and later transitioned to the agile methodology of software development or in situations where both traditional and agile methods are being used.

1.3 Research question

*What is the impact of the use of agile methodologies on software quality criteria compared to traditional methodologies**?***

By seeking to determine the above impact or association, the study will also implicitly or indirectly ascertain whether or not selected software quality criteria can be linked to methodology use in general.

The main research question articulated above is then disintegrated into the following two sub-questions, which in turn, have sub-questions as well:

Sub-question 1

What is the impact of the use of agile methodologies on system quality criteria compared to traditional methodologies?

This first sub-question is further broken down into the following sub-questions, seeking to understand impact of the use of agile methodologies compared to traditional methodologies, on:

1.3.1 Ease of system maintenance;
1.3.2 System testing;
1.3.3 Ease of system training;
1.3.4 Ease of system learning;
1.3.5 System architecture;
1.3.6 Portability;
1.3.7 Meeting usability needs;
1.3.8 Ease of system customisation;
1.3.9 Ease of system navigation;
1.3.10 Ease system interactivity;
1.3.11 System correctness;
1.3.12 Timeliness; and
1.3.13 Completeness of system features.

Sub-question 2

What is the impact of the use of agile methodologies on information quality criteria compared to traditional methodologies?

This second sub-question is further broken down into the following sub-questions, seeking to understand impact of the use of agile methodologies compared to traditional methodologies, on:

1.3.14 Error message comprehensibility; and

1.3.15 System help facilities

1.4 Research objectives

The objective of this study is to investigate if a positive correlation exists between system and information quality criteria when agile methodologies are used compared to traditional methodologies.

Similar to the just stated research question, the main objective will be broken down into two sub-objectives, which in turn have their own sub-objectives.

Sub objective 1

To ascertain whether a positive correlation exist between the following system quality criteria when agile methodologies are used compared to traditional methodologies:

1.4.1. Ease of system maintenance;

1.4.2. Ease of system testing;

1.4.3. Ease of system training;

1.4.4. Ease of system learning;

1.4.5. Robustness of system architecture;

1.4.6. Portability;

1.4.7. Meet usability needs;

1.4.8. Ease of system customisation;

1.4.9. Ease of system navigation;

1.4.10. Ease of system interactivity;

1.4.11. System correctness;

1.4.12. Timeliness; and

1.4.13. Completeness of system features.

Sub-objective 2

To ascertain if a positive correlation exists between the following information quality criteria when agile methodologies are used compared to traditional methodologies:

1.4.14. Error messages comprehensibility; and

1.4.15. System help facilities

1.5 Research motivation and relevance

A review of the research literature on agile methodologies and software engineering in general reveals that most of the studies concentrate on the development process, rather than the product. Research activities in software engineering and information technology (IT) are twofold; build and evaluate. Build refers to the construction of the product to demonstrate that it can be built, and evaluate means the development of criteria to assess the performance of the product in its context of application (March and Smith, 1995).

A number of system development methodologies have been proposed since the inception of the computing discipline, but there is no methodology that integrates empirical evaluation or assessment of the methodology itself, as well as the computer-based artefact that it instantiates in its context of use, so that we can compare beliefs or claims with what we see. Evidence on this was uncovered by Perry et al. (2000), who emphasises the need for empirical assessment of computer systems. Similarly, Zhang et al. (2010) call for more empirical studies on agile methods, as more organisations continue to adopt and use them. Dyba and Dingsoyr (2008a) conducted a systematic review on agile software development, where they found that out of 1996 studies, only 36 were identified as empirical. A paradigm shift is needed to not only concentrate on the agile development process, but also to direct research efforts towards evaluating artefacts constructed using agile methodologies (Abrahamsson, Conboy and Wang, 2009; Bhasin, 2012).

Therefore, considering the fact that research in this domain cannot be left to speculations and secondly, considering the utmost importance, indispensable and strategic role played by information systems in contemporary society, the researcher can rightfully assert that this study and the dissemination of its findings will be relevant to scholars, researchers and practitioners alike.

1.6 Research design and methodology

This research is based on positivism, which posits that the world exists externally, and can be objectively measured (Knox, 2004). It is associated with the Scientific method, which assumes that firstly, the world is structured, regular and not random, and secondly, that the world can be investigated objectively (Oates, 2006).

The chosen research strategy for this study was a survey. The idea behind a survey is to collect the same kind of data in a standardised and systematic way, from a representative sample of the population, and thereafter, to look for patterns and trends to draw conclusions and make generalisations that extend to the wider population (Oates, 2006). Surveys are very useful in collecting data about behavioural aspects that are difficult to observe (Abbas et al., 2010). They

are one of the main research strategies used in IS research. This strategy was chosen because it is cost effective, can lead to generalisations, the data can be replicated to some extent, surveys lend themselves to quantitative data analysis, and finally, because the research questions can conveniently or appropriately be answered with a survey (Oates, 2006).

Data was collected by designing a structured questionnaire that addresses the variables to be investigated. The sample frame was IT practitioners, comprising mainly, but not exclusively, project managers, business analysts, system analysts, architects, developers, product managers, software test analysts, consultants and business stakeholders, who have worked in both agile and traditional environments.

The type of statistical analysis undertaken was descriptive and correlational. Descriptive statistics techniques are normally used to describe and compare variables numerically, while correlational analysis techniques are employed to find relationships between variables (Saunders et al., 2009).

In addition, graphical models were also used perform categorical analysis. They are normally used to visually represent data and show comparisons between categorical data or to represent the number of observations in a given category.

1.7 Definition of Key terms and concepts

The purpose of this section is to define, clarify and operationalise key terminology and research variables under investigation.

1.7.1 Definition of agility

Agility refers to the eradication of much of the bureaucracy that characterises traditional approaches to software development. Agility strives to promote quick responses to the dynamic environment, to welcome changes in user requirements, and to speed up project timelines (Erickson, Lyytinen and Siau, 2005). It can thus be inferred that agile methodologies embrace and reiterate speed and flexibility in the development process, as opposed to following a rigid plan. They focus on iterative and incremental development, so as to deliver value to the customer as quickly as possible. In other words, agile development methodologies make use of iterative development, frequent collaboration with the customer, small frequent releases, and code that has undergone rigorous testing (Cao, Mohan, Xu and Ramesh, 2009). In addition to these definitions, the working definition of agility for this study will be the core values, principles and underlying philosophy as per Agile Manifesto.

1.7.2 System and information quality definition

Quality means different things to different people. Juran, who is widely considered to be a quality guru, defined quality as fitness for a given purpose, accompanied by customer satisfaction (Juran, 1992). Weinberg defines quality as the value something has to a given set of people (Weinberg, 1992). As alluded to in (Godbole, 2004) ISO 9126 defines software quality as the features of a product that have the ability to satisfy certain needs.

Taking cognisance of the different and diverse criteria for software quality, Delone and McLean (1992) fleshed out a dichotomy from the word software quality into system and information quality, in their influential and empirically validated information system (IS) success model (Delone and McLean, 2003). The splitting is depicted in Figure 1.

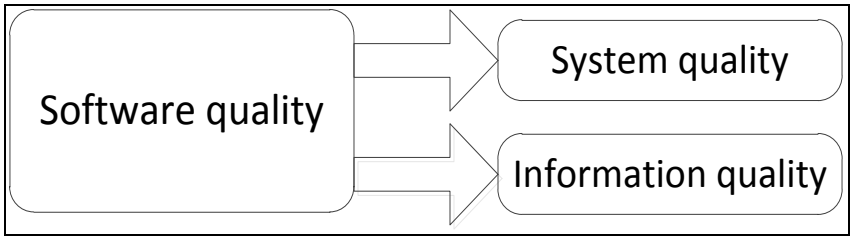

Figure 1 : Software quality dichotomy model.

System quality refers to measures of the information processing system itself (Delone and McLean, 1992; 2003). This refers to a collection of features that attempt to measure the system itself, and typically consist of ease of use aspects and performance characteristics of the system under evaluation (Urbach and Müller, 2012).

Information quality refers to measures of the information system output (Delone and McLean, 1992; 2003). This focuses on measures related to the quality of the information that the system produces, and their usefulness to the user (Urbach and Müller, 2012).

From the above, it is apparent that quality is therefore an elusive concept, and proves difficult to define. A consensus and all-encompassing definition is difficult to arrive at, because different authors define it differently.

This therefore means there are numerous and multiple subjective measures of quality. However, there are some common quality parameters that can be deduced from most definitions. This study will thus base itself on relevant system and information quality criteria set out by other researchers.

1.7.3 Scope and software quality criteria

1.7.3.1 System quality criteria

Several measures of system quality exist. It is therefore not possible to examine all of them in this study, due to time constraints and research scope. The scope will be limited to System and Information quality criteria identified in Tables 2 and 3. Most of them are exemplary measures that have been validated (Urbach and Müller, 2012).

#	Criteria	Definition	Reference
1.	Ease of system maintenance.	Ease of modifying the system to correct defects or new functionality to meet new requirements.	Mnkandla and Dwolatzky (2006).
2.	Ease of testing.	Refers to ease of testing system functionality to uncover defects.	
3.	Ease of system training.	Ability to easily train prospective system users.	
4.	Ease of system learning.	Ability to easily learn the system by prospective system users.	Gable et al. (2008), Sedera and Gable (2004b).
5.	Robustness of the system architecture.	Ability of the system to act appropriately under abnormal conditions as well the ability for the system to withstand extensions with the addition of other components without failing.	Mnkandla and Dwolatzky (2006).
6.	Portability.	Refers to ability of the software to be easily installed on different hardware and software platforms.	Mnkandla and Dwolatzky (2006).
7.	Meet usability needs.	This refers to how easily the system can be used by users of different backgrounds.	Gable, Sedera and Chan, (2008); Mnkandla and Dwolatzky (2006).
8.	System correctness.	Ability of the system to function according to the specification or how the client wants.	Sedera and Gable (2004b).
9.	Ease of customization.	Ease with which the system can easily be adapted to suit one's personal approach and preferences.	Gable et al. (2008), Sedera and Gable (2004b).
10.	Ease of system navigation.	Ease of moving around the system from one page or functionality to another.	McKinney et al. (2002).
11.	Ease of system interactivity.	The degree of interactivity of the system i.e. the extent of reciprocal actions between the user and the system.	McKinney, Kanghyu and Zahedi (2002).

12.	Timeliness.	Refers to timely delivery of software to the market or client.	Mnkandla and Dwolatzky (2006).
13.	Completeness of system features.	Refers to whether a system is delivered with all appropriate parts or components.	

Table 2 : Exemplary system quality criteria.

1.7.3.2 Information quality criteria

#	Criteria	Definition	Reference
14.	Error messages Comprehensibility.	Refers to the ease of understanding error messages.	
15.	System help facilities.	Refers to a subset of the software component that acts as documentation for the software package. Its main purpose is to explain program function, toolbar option, or other key elements with the user interface.	

Table 3 : Exemplary Information quality criteria.

1.8 Outline of dissertation chapters

This dissertation is demarcated into five chapters, namely: an introduction and background, literature review, research design and methodology, data analysis and interpretation of results and culminates in a conclusion chapter.

Chapter 1. Introduction and background: This chapter contextualises the study by presenting a general background. It also succinctly articulates the problem statement, the research questions and objectives, motivation and rationale for the research, a brief study design and methodology that was employed, a definition of key terms and concepts, and finally, the dissertation structure.

Chapter 2. Literature review: This chapter deploys a critical commentary and extensive literature review relevant to the study, with the objective of identifying a gap, weaknesses or flaws in the already existing body of knowledge on agile methodologies, from a software quality enhancing perspective.

Chapter 3. Research design and methodology: This chapter articulates and justifies the research design and methodology that was employed. It elaborates on the tools or instruments that were utilised and procedures that were followed.

Chapter 4. Data analysis and interpretation of results: This chapter is an articulation of data analysis methods and techniques employed; it also unpacks and interprets results of the study and finally discusses and disseminates findings.

Chapter 5. Conclusion: This chapter is a synopsis of previous chapters as well as research findings. It highlights with evidence that the research question was answered and objectives met. It also articulates the research significance and contributions. Like any other research, limitations are acknowledged. Finally, based on research findings and gaps in the existing literature, recommendations for future research areas are proposed.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

The purpose of this chapter is to review relevant literature on the topic under research. It will not only extend the conceptual foundations already laid in Chapter one, but will also help to guide the research process, as well as provide a theoretical framework for analysis and interpretation of findings. The first section is an overview of traditional methodologies and current trends. The second section is an overview of agile methodologies, an in-depth discussion of current trends, and a brief articulation of the theoretical underpinnings of agile methodologies from two contradictory perspectives. The third section is a critical review on related work, done by other researchers. The fourth section is a high level review of software quality models or standards. General trends and observations identified from literature, with a particular focus on the adoption of agile methods are unpacked in the fifth section. Concluding remarks are then made.

2.2 An overview of traditional methodologies

Traditional methodologies of software development are discipline and systematic (Nerur, Mahapatra, Mangalaraj, 2005; Jiang and Eberlein, 2009; Saxena and Kaushik, 2013). They are based on a series of sequential phases, from feasibility study, requirement elicitation, through to design, coding and testing to solution deployment and maintenance. In addition, they also make use of enormous amounts of documentation, and are therefore classified as heavy-weight methodologies (Nikiforova, Nikulsins and Sukovskis, 2009; Mohammad, Alwada'n, Ababneh and Jordan, 2013). As a complement to the forgone definition, traditional methodologies for the purpose of this research also refers to any methodology that has been used by the Software engineering community prior to the advent or emergence of agile methodologies in 2001.

2.2.1 Traditional methodologies summary matrix

There has been a tremendous increase or proliferation of software development methodologies since the inception of the computing discipline (Jayaratna, 1994). The proliferation of these methodologies has led to confusion in selecting an appropriate methodology for a given situation in a non-subjective way (Siau and Rossi, 1998; Avison and Fitzgerald, 2006). According to Jayaratna (1994), about 1000 methodologies exist. The approximate number must have increased by now. However, this section is not intended to present an exhaustive and comprehensive coverage of all of these, but rather to review the ones that have been dominant and commonly-used in the Software development process over the years. Dominance and popularity amongst software professionals and industry were used as criteria for selection. The Waterfall Model, the Spiral Model, the V-Model and the Rationale Unified Process are some of

the most common and dominant traditional methodologies (Zhang et al., 2010; Leau, Loo, Tham and Tan, 2012) and will be reviewed at a high level in Table 4. A review of these methodologies serves two main purposes; firstly, it further clarifies the context of this study, and secondly serves as an indispensable reference and theoretical framework for the comprehension of future chapters and this study in general.

#	**Traditional methodology**	**Description**	**Reference**
1.	The Waterfall Model.	This is one of the oldest software engineering methods postulated by Royce in 1970. It is document-driven and made up of sequential phases from analysis, design, coding, testing and implementation. It has well-defined boundaries, and is thoroughly documented, with documents generated in previous phases serving as input to the next phase.	Royce 1970; Sommerville 2007; Schach 2010.
2.	The Spiral Model.	Originally postulated in1986 and refined in 1988 by Barry Boehm. Risk is inherent in any development initiative, therefore the Spiral Model is an iterative risk-based approach that makes use of a proof of concept prototype to test feasibility and mitigate risk during each phase of the project. The model is spiral in nature, with the radial dimensions representing cumulative cost and angular dimensions representing progress.	Boehm 1988; Schach 2010.
3.	The V-Model.	This model is similar to the waterfall model. The process steps do not flow down in a linear fashion but bent upward to form a typical V. The Model depicts the relationships between each phase of the development life cycle and its corresponding phase of testing.	Sommerville 2007; Zhang et al., 2010.

4.	The rationale unified process (RUP).	This is an object-oriented methodology originated by Rational Software and now owned by IBM. Unlike the Waterfall Model which is one-dimensional, it is a two-dimensional model, with the vertical dimension representing the workflows, beginning with the requirements workflow, through to the implementation workflow, and the horizontal dimension representing the phases starting from inception, elaboration, construction and transition.	Schach 2010.

Table 4 : Traditional methodology summary matrix.

2.2.2 Current state of traditional methodologies

The key characteristics of traditional methodologies were briefly articulated in Chapter one, however, the main overarching feature inherent in traditional methodologies is that they are plan-based, rigid, and focus on detailed documentation (Sommerville 2007; Khan, Qurashi and Khan, 2011; Saxena and Kaushik, 2013).The current literature shows these still to be the dominant methodologies in use, even though some organisations have started to changeover to agile methods (Zhang et al., 2010; Kendall, Kong and Kendall 2012; Moniruzzaman and Hossain, 2013).

Several literature sources assert that the dynamics of each development project is unique, where no methodology proves itself to be a panacea for all software development problems (Schach, 2010). Moreover, there are certain projects where traditional methodologies are found to be most suitable, whereas agile methods are most suitable in others. For example, traditional methodologies are suitable in circumstances where requirements are largely known and fairly stable, whereas agile methodologies are suitable in situations where requirements are largely emergent and frequently change (Boehm and Turner, 2004; Schach, 2010; Aitken and Ilango, 2013; Kumar and Bhatia, 2014). Given the uniqueness of software development projects, several authors have therefore recommended that it makes sense to identify and incorporate the best features of each methodology in any particular software development context (Khan, Qurashi and Khan, 2011; Aitken and Ilango, 2013; Saxena and Kaushik, 2013 Kumar and Bhatia, 2014). From the above-mentioned review, one can then infer and predict that traditional methodologies still have an indefinite life span for the time being and also for the foreseeable future.

2.3 An overview of agile methodologies

Agile methodologies were created and promulgated in 2001 by a group of experience software professionals. They asserted that their methodologies are a better way to develop software (Cunningham, 2001). They went further to create a manifesto that details the precise emphasis of agile development. Explicitly articulated in the agile manifesto was the view that individuals, interactions, working software, customer collaboration, and responding to change, which are all agile values, are more valued than processes, tools, comprehensive documentation, contract negotiation and following a plan, which are all traditional values. Twelve principles were then formulated out of the above-mentioned agile values to form the basis and philosophy of agile development (Cunningham, 2001). The following four main factors underpin agile development: early customer involvement; iterative development; self-organising teams; and adaptation to change (Cunningham, 2001; Saxena and Kaushik, 2013).

Agile methodologies are therefore based on the idea of developing software in an incremental and iterative fashion, whereby phases of the life cycle are revisited over and over using customer feedback to ultimately deliver the solution (Szalvay 2004; Sommerville, 2007; Saxena and Kaushik, 2013). The rationale behind agile methods is to develop high quality software in a cost-effective and timely way, while simultaneously meeting changing customer needs (Zhang et al., 2010).

2.3.1 Agile methodologies summary matrix

Unlike traditional methodologies, agile methodologies are relatively new and emerged at the beginning of the current millennium (Cunningham, 2001). This section is not intended to give an exhaustive and comprehensive coverage of all them, but rather to review the ones that have been dominant and have been commonly used in the software development process since the inception and promulgation of the Agile Manifesto in 2001. Similar to traditional methodology selection in section 2.2.1, dominance and popularity amongst software professionals and industry were used as criteria for selection. Extreme programming, Scrum, Lean Development, Feature-Driven Development, Dynamic Software Development Method, Crystal Methodologies and Adaptive Software Development are some of the most commonly used and dominant of the agile methodologies (Dyba and. Dingsoyr, 2008; Rao, Naidu and Chakka, 2011; Pathak and Saha, 2013) and will be reviewed at a high level in Table 5.

#	Agile methodology	Description	Reference
1.	Extreme Programming (XP).	XP is one of the first agile methods proposed by Kent Beck. Its main emphasis is on best practices of software development that, when put together, will form a successful practice. The following twelve principles define XP: the planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective ownership, continuous integration, a forty hour week, on-site customers, and coding standards.	Beck 2000; 2003.
2.	Scrum.	Focuses on small teams working together to develop software, with an emphasis on feedback mechanisms. Releases software incrementally, called sprints. Scrum is governed by three primary roles, called product owners, development teams and Scrum master.	Schwaber and Beedle, 2001.
3.	Lean Development.	This methodology originated from the manufacturing industry and has been adapted to accommodate software development. It has seven core principles: the elimination of waste, the amplification of learning, decide as late as possible, delivering as fast as possible, empowering the team, building integrity, and perceiving the whole.	Poppendieck and Poppendieck, 2003.
4.	Feature-Driven Development.	Commences by designing an overall model and delivers software by features. Claims to be suitable for critical projects. Its life cycle consists of the following phases: developing the overall model; building a list of features;	Palmer and Felsing, 2002.

		planning by feature; designing by feature; and building by feature.	
5.	Dynamic Software Development Method (DSDM).	Partitions a project in three phases: pre-project, project life-cycle, and post project. The method is underpinned by the following nine principles: active user involvement, empowering the project team to make decisions, focusing on frequent delivery, addressing current business needs, iterative and incremental development, allowing for reversing changes, high-level scope being fixed before project starts, testing throughout the lifecycle, and efficient and effective communication.	Stapleton, 2003.
6.	Crystal Methodologies.	It is a family of human-powered, adaptive, software development methodologies. The following seven principles underlie the method: Frequent delivery, reflective improvement, close or osmotic communication, personal safety, focus, easy access to expert users, technical environment with automated tests, configuration management, and frequent integration.	Cockburn, 2004.
7.	Adaptive Software Development.	An adaptive methodology underpinned by the following five principles: adaptive culture, adaptive frameworks, adaptive collaboration, adaptive scale, and adaptive management. It is a method that views and accepts continuous change as normative.	Highsmith, 2004.

Table 5 : Agile methodology summary matrix.

2.3.2 Current state of agile methodology adoption

Agile methodologies represent a major deviation from conventional, plan-based approaches to software development (Highsmith, 2002; Dyba and Dingsoyr, 2008a; Bhasin, 2012). They all emphasise the idea of flexible, speedy, incremental and iterative development (Larman and Basili, 2003; Larman, 2004; Zhang et al., 2010; Rao, Naidu and Chakka, 2011; Saxena and Kaushik, 2013).

The current literature shows that agile methods are increasingly becoming more popular as the number of organisations shifting to agile methods increases (Schwaber, Leganza and D'Silva, 2007; Salo and Abrahamsson, 2008; Zhang et al., 2010; Esfahani, Yu, and Annosi, 2010; Laanti, Salo and Abrahamsson, 2011).

Despite the growing number of organisations shifting to agile, several studies have found that there are many problems, impediments and difficulties encountered with agile adoption as well as putting the proposed benefits into practice (Barlow et al., 2011; Leau, Loo, Tham and Tan, 2012; Prause and Durdik, 2012; Asnawi, Gravell and Wills, 2012; Pathak and Saha, 2013; Twidale and Nichols, 2013). Some organisations adopt these methodologies to gain a competitive advantage, while others are still sceptical as to whether agile development is beneficial (Barlow et al., 2011), or as to whether the benefits of agile outweigh the cost (Ambler, 2008; Selic, 2009).

The just discussed contradictory perspectives about agile adoption are an interesting trend that remains to be monitored. From the researcher's viewpoint the complete phasing out of traditional methods by agile ones seems unlikely. One can only predict with some degree of certainty a compromise situation, where the two methods will complement one another.

2.3.3 Theoretical underpinnings of agile methodologies

Several authors have attempted to trace the roots of agile methods, with some asserting that their principles and practices have been in software development since the 1960s (Merisalo-Rantanen, Tuure and Matti, 2005). Similarly, Meso and Jain, (2006) also found that agile practices have been previously employed in complex adaptive systems.

Conboy and Fitzgerald (2004) described agility as comparable to flexibility and leanness in other disciplines, by referring to agile manufacturing and Lean development as examples. Jian and Eberlein (2009) have asserted that the idea of incremental development and delivery employed by most agile methods can be traced back to the 1930s, when a quality expert used such a method at Bell labs to enhance product quality (Larman and Basili, 2003). Gladden and Gilb had

long proposed the practice of delivering working software early, to address the late delivery phenomenon that leaves the customer dissatisfied (Gladden, 1982; Gilb, 1985).

Jian and Eberlein (2009) conducted a study on the historical links of traditional and agile methodologies from three perspectives, namely; practices, principles and technological contexts. They concluded that there is significant evidence that practices used in both traditional and agile methodologies have historical links and that many practices in both methodologies have roots in other disciplines, as well as in traditional engineering.

However, despite the forgone commentary, sceptics argue that the Agile Manifesto principles are insufficiently grounded in theory (Conboy and Fitzgerald, 2004), particularly management theory and philosophy. Furthermore, they assert that a universally accepted definition of agility is non-existent and that given such vagueness and diverse interpretations, it becomes a daunting task trying to draw meaningful conclusions from their use. Hummel (2014) conducted a systematic review on agile methodologies and concurred with the view that an agreed upon definition of agility is not readily observable. Aitken and Ilango (2013) also agree that there is no clear or agreed upon definition of the agile paradigm. Hummel (2014) further advised that research based definitions should be used in order to enable better comparison of studies, instead of referring to the Agile Manifesto (as most authors do), which consists of untested principles and practices with no solid theoretical basis. Dingsoyr, Nerur, Balijepally and Moe (2012) also concluded after their study on agile methodologies that they lack theoretical underpinnings, adding that not enough attention is being paid to the theoretical foundations of agile when investigating them. They then urged agile researchers to embrace a theory-based approach in their research agendas. They argued that theory-driven research helps to separate true innovations from reinventions and remixes of old approaches.

2.4 Related work and critical review

Some related studies have reported that agile methodologies do improve product quality, although some studies have been inconclusive, and others have been contradictory (Dyba and Dingsoyr, 2008). With this contradiction of viewpoints, controversy, turmoil and speculation regarding the perceived benefits of agile development, a case to further research agile methodologies is built because their creation and adoption has largely been driven by consultants and practitioners. (Abrahamsson, Conboy and Wang, 2009; Hummel, 2014).

A comparative analysis between the Waterfall model and Agile methodologies by Huo, Verner, Zhu and Babar (2004) concluded that firstly, agile methods do have quality assurance (QA) capabilities built into the development process, and secondly, that the frequency with which these

QA activities occur are more frequent than in the Waterfall Model, and lastly, that these activities do occur very early in the development cycle. This was merely an analysis and subjective opinion of the authors, and was never an empirical study.

Bhasin (2012) undertook a study seeking to understand the relation between agile methods and software quality, in terms of the development factors of code quality, technical debt, defect level, stability; and the customer factors of cycle time, defect density, outage reduction and customer satisfaction index. At the end of the paper, a conclusion was drawn that agile methods have built-in quality management capabilities. It was not apparent how this conclusion was arrived at, or what it was based on, because the study did not provide any data. Secondly, there was no mention of the quality factors that the study sought to investigate.

Mnkandla and Dwolatzky (2006) also conducted a similar study, introducing an innovative technique in order to determine which factors of software quality are improved by the use of agile methods. They identified certain agile activities and linked them to be responsible for specific software quality attributes. The tool was validated using Extreme Programming and Lean development as agile methods. Their findings suggested that agile methodologies do improve software quality with respect to the following parameters: compatibility, cost-effectiveness, ease of use, correctness, efficiency, extendibility, integrity, maintainability, portability, reusability, robustness, timeliness, verification and validation. This study was merely an expert analysis, evaluation and contribution. Generalisations cannot be made, because no rigorous research method was employed.

Ahmed, Ahmad, Ehsan, Mirza and Sarwar (2010) concluded after a quantitative study that knowledge sharing, active stakeholder participation, self-organising teams, reduced documentation, responding to change, team size, flexible designs and training are some of the key attributes of agile methods that impact positively on productivity and the quality of the finished product. They turn around and argued that enough documentation should be available for future maintenance purposes. This is a counter argument for one of the core values of agile advocates, who argue that the creation of documentation should be minimised. Furthermore, the study did not acknowledge the limitations of the research, and particularly, did not address how they attempted to isolate or mitigate other factors that might have distorted the study outcome. Moreover, the specific quality criteria of software that were improved also remained unmentioned. In addition, the inclusion of ‘training’ of professionals as an agile attribute in the study is both surprising and concerning, because it is not exclusively an agile construct. It is a

well-known fact that training of professionals always has a positive correlation with productivity and quality.

In an agile conference research paper by Abbas, Gravell and Wills (2010), the authors concluded that organisations with more agile experience are doing better with regards to project success and code quality. This suggests that benefits are only realised when organisations have gained enough maturity using agile methods. Moreover, the definition of project success was never provided. It was left for the respondents and the reader to interpret what constitutes project success, making the outcome of the research doubtful.

From a code quality perspective, the above study by Abbas et al. (2010) is similar to a quantitative study undertaken by Santos et al. (2011), who concluded that agile methods can contribute to quality in three aspects: greater staff involvement; agile management of requirements proposed; and the code developed. Further to this, Rao et al. (2011) noted that developers stressed the following reasons for adopting agile methods: adaptability to change, short time frame of releases, continuous feedback from customer, high quality and fault free software.

The forgone two studies (Abbas et al; 2010; Santos et al, 2011) concluded that code quality can improve with the use of agile methods. Code quality is only one software quality attribute, and does not necessarily translate to overall software quality. Software quality is a multifaceted concept that needs to be dissected (Delone and McLean, 1992).

Other studies have produced different results. An empirical study by Bonner, Teng and Nerur (2010) confirmed the proposal that agile development leads developers to believe that they are less complex, much more compatible, and lead to increased benefits. These are the beliefs that influence them to likely accept agile methodologies. This study contrasts with a review of 17 companies, where developer fear of skill deficiency exposure was proposed as one of the difficulties faced when implementing agile methodologies (Pathak and Saha, 2013).

Sfetsos and Stameos (2010) carried out a systematic literature review on 46 empirical studies on agile methodologies. One of their main research objectives was to ascertain the current knowledge on quality issues in agile practices. Their main finding was that pair programming improves code and design quality. A misalignment is observable in their study between the study objectives and findings, because the study objective was to study quality issues in agile methods in general. Conspicuously, the study only concentrated on Pair Programming and Test-Driven Development, which are both XP practices (Beck, 2003). In addition, the specific criteria that was used to ascertain that design quality is improved was never disclosed. ‘Improved design quality’

is so vague an assertion. Moreover, construct validity also comes under the spotlight here, construct validity is concern with whether the researcher is measuring the right variables under study via the questions in the questionnaire or is the study measuring what the researcher thinks it is measuring (Oates, 2006). The fact that the study was supposed to measure agile practices in general in relation to quality but only focused on XP practices means construct validity issues were not well addressed.

Kendall, Kong and Kendall (2012) carried out a study on the impact of agile methodologies on information system (IS) quality, they constructed a conceptual model adapted from Delone and McLean's model of IS success (2003). They disintegrated software quality into system and information quality (Delone and McLean, 1992). The authors found that the use of agile methodologies firstly affects internal performance, and indirectly affected the quality of the software product. Finally; they did not conclude explicitly and concisely that the use of agile methodologies improved system and information quality. This was obscured from their conclusion.

Schmidt, Srinivasa and Heymann (2014) conducted an empirical investigation seeking to understand the perceived benefits of agile software engineering practices. From the research context, two case studies from SAP were used as the research strategy. As with most studies, they also concentrated on pair programming and tested automation as agile practices. Software quality was categorised into external and internal quality (ISO/IEC 9126, 2001). External quality was measured firstly by quality in general and second by reported defects. Internal quality was measured by application programming interface (API) quality, modularity and understandability. They went further to survey about 200 software professionals, mainly developers. They concluded that agile software practices have a positive effect on quality and efficiency of development teams. This study was only carried out on SAP, which is an enterprise resource planning (ERP) system. It was not also carried out on in-house developed systems, or a complex system from scratch, and therefore, generalisations cannot be made. Conspicuously, same as several studies, it only concentrated on XP practices. Furthermore, developer's responses were based on subjective self-assessments of their code, which introduces bias and potentially distort the results. Moreover, most of the variables used in the study, for example, quality in general, API quality, modularity and understandability were not defined, leaving room for ambiguous interpretations.

Dyba and Dingsoyr (2008a) conducted a systematic review of 36 empirical studies on agile software development. Some of the results relating to the quality of the product when agile

methods are used were contradictory. They also found that some of the studies reported an increase in code quality when agile methodologies are used, but concluded that none of the studies had an appropriate recruitment strategy to ensure the comparison was not biased. Still, on the systematic review undertaken by Dyba and Dingsoyr (2008a) on empirical studies of agile software development, they frequently found that methods were not well or properly described, validity, reliability and bias issues were not always addressed, data collection methods and analysis were not explained well. None of the studies got a full score on the assessment criteria just stated.

Hummel (2014) conducted a systematic literature review in the field of agile information system development, encompassing 482 papers. Building and extending on previous reviews, the author found that most of the research methods employed where qualitative and recommended that more confirmatory quantitative studies are needed to corroborate previous or initial findings. He also uncovered that the theoretical underpinnings of agile methodologies are lacking.

With this study, the researcher did everything possible to employ a structured and rigorous research methodology. Validity, reliability and bias issues have been addressed to the best ability of the researcher, though it is not practically possible to address all of them. The data collection methods and analysis have been explained in a way that the reader can audit the process for transparency, objectivity and credibility.

While acknowledging and paying credit to previous related research, the following shortcomings, or gaps were summarily identified in the current literature; firstly, empirical studies that concluded that the use of agile methodologies leads to improved product quality did not explicitly state which software quality metrics were used. Secondly, the reviewed studies were mostly carried out in small isolated settings, and no global study was found at the time of writing this dissertation. Thirdly, the software quality attributes that this study has employed as the dependent variables have not been used before with agile and traditional methodologies as the independent variables. Dependent and independent variables in this research are mathematically defined by the following equation:

Agile or traditional methodologies (Independent variables) = Software quality parameters (Dependent variables).

This study has focused on the above discussed gaps and limitations with previous studies in the following ways: firstly, the software quality metrics that have been used to investigate their relation with agile and traditional methodologies have been identified and defined. Secondly, it

has been carried out on a local and global scale, and lastly, at the time of writing this dissertation, no prior study of this nature was found.

2.5 High-level review of software quality models

Since this study is about software quality, the researcher saw it worthwhile or necessary to include a brief high level review on software quality models and standards. The proliferation of software quality models or measures has been ubiquitous. All these models broadly fall into one of the main categories discussed in the next paragraphs even though hybrid variants exist (Ferenc, Hegedus and Gyimothy, 2014).

2.5.1 Software process quality models

This category focuses on the development process rather than the product. The rationale behind these models is to measure and improve the software development process with a goal of ultimately improving the quality of the product (Schach 2010; Ferenc et al.,2014). Typical and popular examples include Capability Maturity Model (Chrissis, Konrad and Shrum, 2003), Project Management Maturity Model (Farrokh and Mansur 2013) and ISO 9000 series (Schach,2010).

These models have the following shortcomings; firstly, most maturity models are staged models in the sense that their adopters will need to progress from one maturity stage to another satisfactorily in order to reach an optimal maturity level. This takes time and does not happen overnight (Schach 2010, Farrokh and Mansur 2013). Secondly, most of the models have definite key performance indicators(KPIs) that organisations need to improve upon. This makes them not flexible or scalable enough for organisations that wish to improve only a subset of their processes (Farrokh and Mansur 2013).

2.5.2 Software product quality models

This category of models focuses on the product. These class of models measure the software product itself. Early examples of this kind of models include the McCall's quality model (McCall, Richards and Walters,1977) Boehm's quality model (Boehm, Brown, Kaspar and Lipow,1978) and Dromey's quality model (Dromey 1995). The main challenge with these models was that there were too theoretical by design and complicated to practically apply (Ferenc et al., 2014).

The next subset of models focusing on the product are called Metrics-based Empirical Prediction models. These sub-class of models are much more practical relative to the McCall, Boehm and Dromey's early quality models and use software metrics as quality predictors (Ferenc et al., 2014). Typical examples include the Univariate Linear Regression Analysis model (Zhou and Xu

,2008) and the Bayesian Network model (Koten and Gray,2006). One major drawback with these models is that they only focus and use maintainability as a quality predictor (Ferenc et al., 2014).
The final subcategory of models focusing on the product are called Practical quality models. These are models that are directly applicable for evaluating the quality of software systems (Ferenc et al., 2014).
Typical examples include the Quamoco quality model (Wagner et al., 2012) and the SQUALE quality model (Mordal, Anquetil, and Laval, 2013).
The major weakness identified with all the quality models reviewed in this section is that their correctness and accuracy has not been validated (Singh and Kannojia,2013; Ferenc et al., 2014). Further research is therefore needed in this area.

2.6 Trends and observations identified from literature

From the prior review, there is ample evidence that there is still a great deal of pandemonium, scepticism and contradiction regarding the perceived benefits after the adoption of agile methodologies. Most studies have reported that agile methods work very well in small environments or projects (Dyba and Dingsoyr, 2008a; Barlow et al., 2011; Saxena and Kaushik, 2013), but empirical evidence of their success in general and in large environments remains scarce (Abrahamsson, Oza and Siponen, 2010; Barlow et al., 2011). Furthermore, it was also observed that most of the studies are carried out in small, isolated settings, in the form of case studies or surveys (Barlow et al., 2011). Very few studies have been carried on a large scale (Barlow et al., 2011), and no global study was found at the time of writing this review raising questions as to the external validity of results. Another glaring observation was that studies focusing on software quality tend not to explicitly state the quality criteria used or use different quality criteria, making comparisons difficult. This is understandable, given the implicit ambiguity of the concept of quality. Another trend identified was that most of the studies tend to focus on few familiar agile methods, which are Extreme programming and Scrum (Dyba and Dingsoyr, 2008a; Dingsoyr et al., 2012; Hummel, 2014) and at the end all-encompassing and broad conclusions are made (Laanti, Salo and Abrahamsson, 2011). In addition, evidence from literature showed that existing software quality models have not been validated for correctness and accuracy. Lastly, some of the conclusions are unclear and end up not saying anything concrete, or use obscurant or arcane language, making it difficult for the reader to actually discern the research outcome.

2.7 Conclusion

This chapter reviewed relevant literature on traditional and agile methodologies, highlighting current trends. The theoretical foundations of agile methods were also briefly reviewed, from two conflicting views. Limitations, flaws, gaps and shortcomings in prior related work were uncovered. The gap that this study will close, or how it will fit to what has already been done, was also articulated. Current trends and observations showed that most quality-related studies concentrated on three XP practices, namely pair programming, refactoring, and test-driven development (Sfetsos and Stameos, 2010; Dingsoyr et al., 2012; Schmidt, Srinivasa and Heymann, 2014). Secondly, quality criteria used in several studies are not explicitly stated, and lastly, most studies are carried out in small isolated settings. From a quality enhancing perspective on agile methodologies, it was apparent that conflicting views still exist, with no apparent consensus having yet emerged. To this end, there is sufficient premise to further research on agile methods.

CHAPTER 3: RESEARCH DESIGN AND METHODOLOGY

3.1 Introduction

Research design and methodology is a systematic blueprint of how the research process will unfold. It encompasses the plan, methods and strategies of inquiry and underlying assumptions that underpin the research process, as well as dissemination of findings. This chapter is made up of four sections. Section one briefly articulates the main research paradigms, purpose, modes of reasoning and approaches. It justifies the rationale for the use of positivism, explanatory research, quantitative approach and deductive mode of reasoning in this study. Section two unpacks the research strategy and data collection method. Section three explicates succinctly how data analysis and interpretation was done. Section four addresses validity issues and ethical considerations followed by concluding remarks.

3.2 Philosophical paradigm

This research is based on positivism, which posits that the world exists externally and can be objectively measured (Knox, 2004). It is associated with the Scientific method, which assumes that firstly, the world is structured, regular and not random, and secondly, that the world can be investigated objectively (Oates, 2006). Therefore, the idea behind this study is to try as much as possible to objectively measure the impact of agile methods when compared to traditional methods on selected software quality parameters. Paradoxically, quality is such an elusive, subjective concept, and difficult to define as already alluded in Chapter one.

A research approach can either be quantitative, qualitative or mixed (Creswell, 2003). With regards to broad research approaches, Newman and Benz (1998) have asserted that current discourse places less emphasis on the distinction between qualitative and qualitative methodologies, but is rather concerned with the way in which studies tend to lie somewhere in a continuum between the two. The authors contend that quantitative and qualitative methods represent an interactive continuum, rather than a dichotomy or bipolar opposites, arguing that the distinction between quantitative and qualitative approaches is an illusion, and can be misleading (Newman and Benz, 1998).

Onwuegbuzie and Leech (2005) also contend that quantitative methods are not necessarily positivist, nor are qualitative methods necessarily hermeneutic (Sieber, 1973; Cook and Reichardt, 1979; Daft, 1983; Miller and Fredericks, 1991).

As alluded to by Creswell (2003), Phillips and Burbules (2000) define a quantitative study as that which develops true and relevant propositions that serve to explain a situation or phenomenon

of concern by making use of variables. In quantitative studies, the researcher postulates relationships between variables in the form of a research question or hypothesis, and either refutes or confirms them after collecting and analysing the data. This study fits the quantitative paradigm perfectly and is therefore quantitative in nature.

However, on the other hand, the qualitative approach involves the study of information systems in a social context, where the unpredictability of human beings comes to bear, meaning that there is an element of interpretivism (another contrasting philosophical paradigm) implicit in such studies. Interprevitism in IS research seeks to investigate the social context of information systems i.e. the social processes involved in their development, how they influence people, and how people in turn influence social processes in their social setting (Oates, 2006). Interpretivist researchers argue that there is no such thing as an objective, real world, or a single truth, and argue that there are multiple subjective realities and that everything in the world including positivism itself is a social construction engineered by researchers over time (Oates, 2006). Interpretivist researchers therefore argue that reality is subjective, and that people perceive it differently.

Positivism thus postulates that reality is given, while interpretivism argues that reality is a social construction. Despite the forgone controversial and contradictory philosophical perspectives, this research adopted the approach of philosophical singularism, namely positivism.

3.2.1 Research purpose

Saunders, Lewis and Thornhill (2009) asserted that the way a research question is framed will lead to descriptive, exploratory or explanatory answers (research). Exploratory research seeks to determine what is unfolding; trying to find new insights and ask questions to evaluate a phenomenon from a new perspective (Robson, 2002). It proves to be a particularly useful means of clarifying ones understanding of a problem. Descriptive research seeks to articulate an accurate profile of persons, events or situations (Robson, 2002). Explanatory research, as another apex of the triangle, seeks to establish relationships between variables. It concentrates on the study of a problem, situation or phenomenon, with the goal of explaining the relationships between variables (Saunders, Lewis and Thornhill, 2009). This research aims to comparatively investigate the impact of agile and traditional methodologies on software quality parameters, and is therefore explanatory in nature.

3.2.2 Research approach

As introduced, two broad research approaches exist, namely quantitative and qualitative. Quantitative research, also known as empirical research strives to answer questions about relationships between independent and dependent variables (Saunders, Lewis and Thornhill, 2009) and is used to collect and analyse data, from which statistical analysis can be performed to interpret the data and draw conclusions (Oates, 2006). Qualitative research, on the other hand, collects data that is analysed using interpretive methods (Saunders, Lewis and Thornhill, 2009). A distinction between the two types of approaches is summarised in Table 6.

#	Quantitative	Qualitative
1.	Based on meanings derived from numbers.	Based on meanings expressed through words.
2.	Numerical and standardised data is collected.	Non-standardised data is collected requiring classification into categories and themes.
3.	Statistics and models are used for analysis of data.	Conceptualisation is used for analysis of data.

Sources: Adapted from Dey (1993); Healey and Rawlinson (1994).

Table 6 : Distinction between quantitative and qualitative research.

This study leans more towards the quantitative end of the interactive continuum between these methods (Newman and Benz, 1998) and was chosen for the following advantages (Oates, 2006):

- Analysis of quantitative data is based on well-established scientific and mathematical techniques.
- Quantitative data can easily be analysed using software applications.
- Quantitative research is suited to generalisations.
- Using standard approaches, quantitative studies can be replicated longitudinally over time to produce comparable findings.
- In addition to the strengths highlighted above, the researcher found it much more convenient and relevant to employ a quantitative approach for this study because of the nature of the research question.

On the contrary, quantitative research presents the following limitations (Oates, 2006):

- There is a tendency to perform more sophisticated statistical analysis on the data using personal computers, without understanding them properly, and at the expense of the original research question.
- The analysis can only be as good as the data initially collected.

- By focusing too much on numbers, non-quantitative aspects of the research may be missed.
- The data is self-reported, and many decisions taken by the researcher can influence the research outcome in a negative way.

3.2.3 Modes of reasoning

Conventionally, two modes of reasoning exist. Deductive and inductive (Mouton, 2005). Deduction, which is associated with the Scientific method, is a form of logic that draws conclusions from other premises or statements that follow from such premises. Simply stated, it draws conclusions from a particular case based on the general, i.e. using a top-down approach (Mouton, 2005). Induction is a form of reasoning that works from the specific to the general, i.e. using a bottom-up approach. With inductive reasoning, conclusions are drawn based on an observation (Mouton, 2005). The mode of reasoning and reasoning assumption or logic underpinning this research is deduction.

3.3 Research strategy

The nature of the research question for this study dictated a correlational as well as a causal comparative design, which could easily be answered with a questionnaire and survey, to meet research objectives. Both correlational and causal research seek to study relationships among variables, but neither allows for the actual manipulations of the variables (Gay, Mills and Airasian, 2009). The key difference between the two is that correlational involves the researcher trying to establish relationships among variables in one group, while causal comparative research needs at least two or more groups (Gay et al., 2009). Causal comparative research is a kind of non-experimental set of methods that seeks to understand cause and effect between different groups (Fraenkel and Wallen,1993), and “involves selecting two or more groups that differ on a particular variable of interest and comparing them on another variable or variables” (Fraenkel and Wallen, p.321).

These distinctions are summarised in Table 7.

#	**Correlational design**	**Causal comparative design**
1.	Does not involve an attempt to understand cause and effect.	Attempts to infer causation.
2.	Two or more variables.	At least one independent variable.
3.	One group.	Two or more groups.

Table 7 : Correlational vs Causal comparative design matrix. Adapted from Gay, Mills and Airasian (2009).

This study seeks to comparatively study the impact of agile and conventional methodologies on selected software quality parameters, and is therefore correlational, as well as causal-comparative in nature, because two groups are involved. These comparison groups (traditional and agile) are, according to Gay et al. (2009), selected because they differ in some aspect, characteristic or experience (i.e. software quality), or the two groups differ in the amount of Characteristic that they share. Furthermore, these groups are already in existence before the researcher arrives, when the alleged cause and effect have already occurred (Gay et al., 2009).

As already mentioned to earlier, a structured questionnaire was developed and the target audience surveyed. Questionnaires are mostly associated with the quantitative approach. The survey strategy is inextricably linked to the philosophical paradigm of positivism (Oates, 2006). Surveys are noted as being particularly useful in collecting data about behavioural aspects that are difficult to observe (Abbas et al., 2010). They are one of the main research strategies used in IS research. The survey strategy was selected for this study because surveys have the following strengths (Oates, 2006):

- Generalisations are likely to be made because a representative sample of the population was employed.
- The researcher found them to be cost effective, making it easy to collect data relative to other strategies.
- The data collected was easily analysed, using statistical computer applications.
- Survey research like this one can be subjected for further testing, leading to corroboration or refutation of findings.
- The nature of the research question dictated a correlational as well as a causal comparative design which could easily be answered with a survey.

Despite these advantages, the following limitations of surveys must be acknowledged (Oates, 2006):

- They tend to focus only on numbers therefore ignoring important non-qualitative data.
- They are weak at establishing the causal relation as experiments do, but nonetheless show associations or correlations.
- Surveys only provide a static picture at a particular point in time, and do not examine on-going dynamic processes.

- With non-self-administered surveys as in the case of this study, the researcher cannot judge the accuracy of the audience responses by observing their body language.
- With online-administered surveys such as this one, the researcher has no opportunity to probe or ask follow up questions to gain new insight.

3.3.1 Data collection method and sampling frame

Data was collected by designing a structured questionnaire that addresses the variables to be investigated. The sample frame was IT practitioners, comprising mainly, but not exclusively, project managers, business analyst, system analyst, architects, developers, product managers, software test analysts, consultants and business stakeholders, who have worked in both agile and traditional environments. Two data collection strategies were employed namely, online placement of questionnaire via a link and distribution via email.

3.3.1.1 First data collection strategy

After seeking permission from the group managers, the survey was firstly placed online via a link, courtesy of SurveyMonkey for a period of three months to three online professional groups.

The first group was an IT business analyst professional community within LinkedIn, a professional networking site. The group has a global presence, with a membership of 65546 members at the time of this study. It is an open community of Business Analysts including other IT professionals from different specialist areas.

The second group was an Agile business analyst professional community within LinkedIn, which is a professional networking site. The agile group has a global presence, with a membership of 24,663 at the time of writing this dissertation. It is open to anyone practicing agile methods, or interested to know what is going on in the agile space, and is made up of IT practitioners and business managers in different specialist areas.

The third group was a Project management office (PMO) group, also within LinkedIn. This group is mostly made up of project managers. It has a global presence with a membership of 72,670 at the time of writing this dissertation.

3.3.1.2 Second data collection strategy

The second data collection strategy was realised by distributing the questionnaire link via email to a group of between 150-200 IT practitioners in South African organisations, comprising mainly but not exclusively, project managers, business analysts, system analysts, architects, developers, product managers, consultants, business stakeholders and software test analysts.

3.3.1.3 Sampling frame

The sampling type used with both data collection strategies was clustered sampling, which uses the fact that instances of the research population of interest might naturally occur together in clusters (Oates, 2006). The first cluster group was the global online IT/Agile business analyst and PMO group. The second cluster group was the IT practitioners in South African organisations that were surveyed.

3.3.2 Questionnaire design

As already mentioned in the prior section, a structured questionnaire was developed and administered to the target audience. The questionnaire was made up of two sections. Section A was meant to capture background information about respondents. Section B was meant to capture data about System and Information quality of a software product.

3.3.3 Key agile and traditional methodologies group characteristics

In causal comparative designs, the groups are selected because they differ in the characteristics they share, or the two groups differ in the characteristics they share to different extents (Fraenkel and Wallen, 1993; Gay et al., 2009; Gall, Gall, and Borg, 2007).This design “involves selecting two or more groups that differ on a particular variable of interest and comparing them on another variable or variables” (Fraenkel and Wallen, p.321).The assertion just stated is analogous to the situation of this study. Agile and traditional methodologies already differ with respect to software quality, and are being compared in this case with respect to other variables (documentation size, system architecture design, customer involvement extent, nature of requirements and project life cycle model).The key distinctions in Table 8 between agile and traditional methodologies, were postulated by leading, prominent and pioneer agile advocates (Highsmith and Cockburn, 2001; Boehm, 2002; Boehm and Turner, 2005; Wysocki, 2009, 2011).

#	Project characteristic	Traditional	Agile	Reference	Applicable Question
1.	Documentation size.	Enormous and detailed documentation produced.	Minimal documentation produced.	(Boehm and Turner, 2005; Pathak and Saha 2013)	15, 16, 17 and 18.
2.	System architecture design.	Design for current and future requirements.	Design for only what is presently essential.	(Boehm, 2002; Wysocki, 2009; 2011)	19 and 20.
3.	Customer involvement extent.	Customer involvement is low and passive.	Customer is onsite, and considered to be team member. Customer involvement is active and proactive.	(Highsmith and Cockburn, 2001; Pathak and Saha 2013)	21,22,23,24 ,25 and 26
4.	Nature of requirements.	Knowable early, largely stable, clearly defined and documented.	Emergent, rapid change, unknown and discovered during the project.	(Boehm, 2002; Boehm and Turner, 2004)	27.
5.	Project life cycle model.	Linear/sequential model.	Evolutionary, incremental and iterative model.	(Charvat, 2003; Nerur,Mahapatra, and Mangalaraj, 2005)	28 and 29.

Table 8 : Key agile and traditional group characteristics.

3.3.4 Correlational and causal comparative design model

A causal comparative design involves the following steps (Johnson and Christensen, 2010):

3.3.4.1 Involves comparing two or more groups on a single endogenous (internal) variable (software quality with this research, ***see step 1 of figure 2***)

3.3.4.2 The key characteristics that distinguish these groups are exogenous (external) variables (project characteristics with this research, ***see step 2 of figure 2***).

3.3.4.3 Groups that differ on exogenous (external) variables of interest are selected (agile and traditional methodologies, ***see step 2 of figure 2***)

3.3.4.4 Thereafter, the groups are compared by looking at an endogenous variable that might be influenced by an exogenous variable (***see step 3 of figure 2***).

3.3.4.5 Next, collect data using a reliable instrument (***see step 4 of figure 2***).

3.3.4.6 Finally, analyse, interpret data and disseminate findings (***see step 5 and 6 of figure 2***).

The design elaborated above is depicted at a high level in Figure 2.

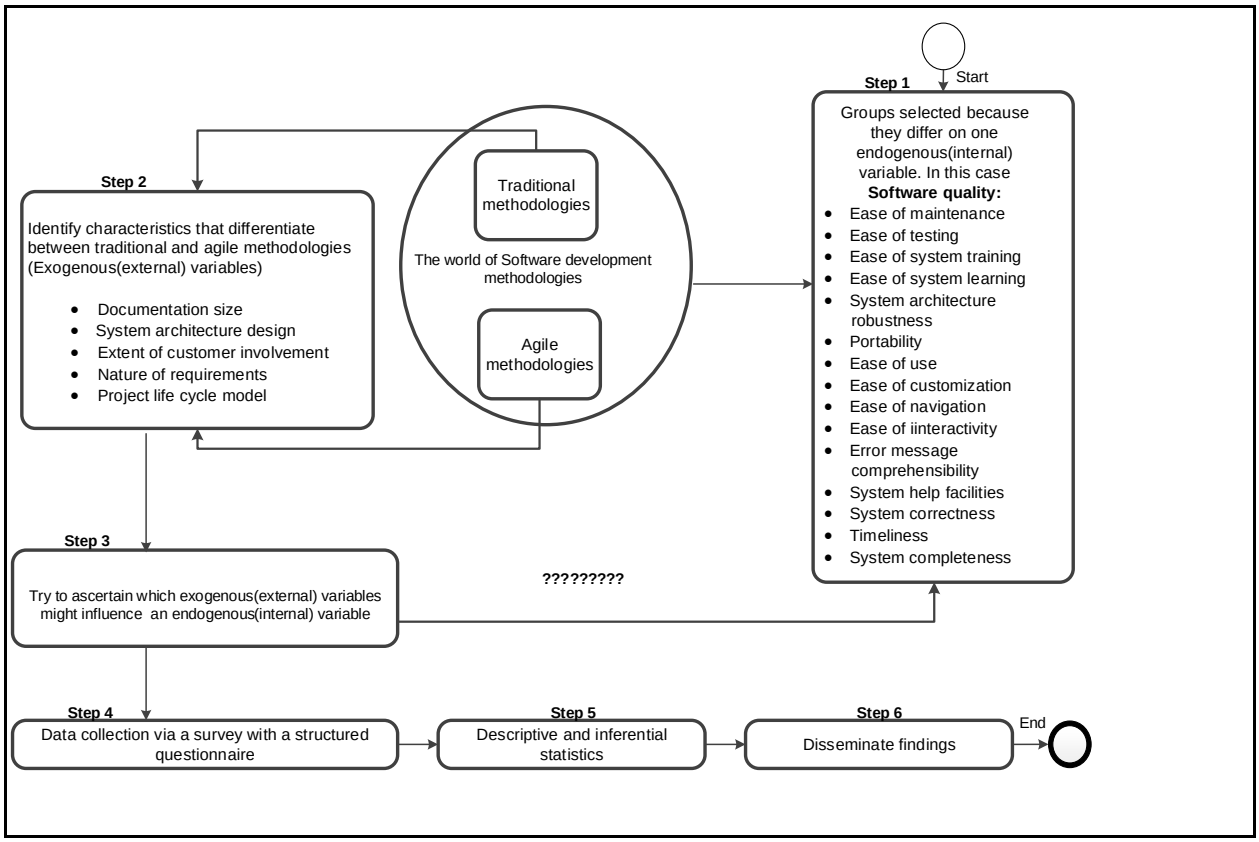

Figure 2 : Correlational and Causal Comparative Design Model.

3.3.5 Question formulation

All questions measuring software quality (see section 7 question15 to 29) were derived by cross tabulating system parameters and project characteristics as informed or underpinned by the causal comparative design (see section **3.3.3** and **3.3.4** for an explanation as well as figure **2**). The question formulation template is depicted in the contingency below (see table 9) followed by the steps taken to actually derive the questions. A contingency table is essentially used to analyse and record the relationship between two or more categorical or nominal variables(Olivier,2009).

System quality parameter under consideration**=A**

B=Software development paradigm	**C=**System developed using traditional methodology	**D=**System developed using agile methodology
E=Project characteristic under consideration	**F=**Characteristic X	**G=**Characteristic Y

Table 9 : Question formulation template matrix

3.3.5.1 Between agile and traditional methodologies, the particular quality criteria that the researcher is interested in is identified, hence **A** gets populated (***see step 1 of figure 2***).

3.3.5.2 Having identified agile and traditional methodologies as two different software development paradigms, cell **B**, **C** and **D** gets populated.

3.3.5.3 The two paradigms mainly differ with respect to project characteristics, hence cell **E** gets populated with the particular project characteristic under consideration eg documentation size or project life cycle model (***see step 2 of figure 2***).

3.3.5.4 Given a particular project characteristic, agile and traditional methodologies differ on this characteristic to different extents, hence cell **F** and **G** gets populated (***see table 8***).

3.3.6 Questionnaire approval

The information and consent note that were released with the questionnaire were reviewed and approved by the Ethical clearance committee of the School of computing at UNISA. It was then piloted and feedback incorporated prior to dissemination to the target audience.

3.4 Data analysis and interpretation

Data analysis is the process of mining the data to find meaning or information, using statistical techniques (Coronel, Morris and Rob, 2009). It is a structured process, which entails looking for patterns, discrepancies, associations and relationships within the data, from which conclusions can be deduced. Owing to the quantitative nature of this study, data capturing and coding, as well as descriptive and inferential statistic techniques were employed to analyse the data.

3.4.1 Capturing and coding

After collecting the data, it was captured in an excel spread sheet using pre-defined codes. The rationale for coding the data is to make it easier to enter in a computer, with fewer errors, so as to be able to perform a quantitative analysis (Oates, 2006). The following guidelines were used to design a coding scheme: codes were mutually exclusive i.e. no overlap between codes; the codes were exhaustive i.e. covering all predefined options; the codes were consistently applied; and codes for missing values were also included (Oates, 2006). Thereafter, the data was checked for mistakes, corrections and cleansing done prior to statistical analysis.

3.4.2 Descriptive statistics

Descriptive statistics techniques were used to compare the variables numerically. Specifically, the following methods were employed (Saunders, Lewis and Thornhill, 2009):

#	Investigation type (inquiry)	Statistical test	Statistical application used
1.	Statistics to measure the central tendency.	Mean and mode.	Statistical Package for the Social Sciences (SPSS).

Table 10 : Descriptive statistics test matrix.

3.4.3 Inferential statistics

Inferential statistics techniques were employed to search relationships, discrepancies and trends. This technique was used in order to reach conclusions that extend beyond the boundaries of the immediate data alone (Saunders, Lewis and Thornhill, 2009). The following inferential statistics techniques were employed (Saunders, Lewis and Thornhill, 2009).

#	Investigation type (inquiry).	Statistical test.	Statistical application used.
1.	To test whether a significant relationship exist between two variables.	Fisher's exact test.	SPSS.
2.	To assess the strength of relationship between two variables.	Cramer's V coefficient.	SPSS.

Table 11 : Inferential statistics test matrix.

3.5 Validity issues and ethical considerations

When a paper is classified as research, it doesn't automatically mean that the results are reliable, valid or generalisable to other contexts. It's important that researchers and practitioners understand the implication of threats to validity in research findings (Valerdi and Davidz, 2009). Despite the advantages of surveys, they also have limitations with regards to validity and reliability. The limitation is specifically related to the data collected, since it is self-reported, and a miscomprehension of the questions by respondents or manipulation of the data by the researcher can propagate inaccuracies in the data (Nardi, 2002). The following paragraphs will identify and explicate the ways in which validity and reliability issues were addressed, including ethical considerations.

3.5.1 Content validity

The issue of content validity often arises when a questionnaire is used as a data generation method. This refers to the ability of the questionnaire to generate data about the variables or concepts under scrutiny (Oates, 2006). All the variables for this study were identified, defined and operationalised. Further to this, all possible measures were employed to design the questions to cover all variables, be brief, relevant, unambiguous and objective. A pilot was

conducted with peers, as advised by Merriam (2002), in order to enhance validity, and feedback was incorporated into the questionnaire. Finally, the questionnaire was included as an appendix at the end of the research report, for scrutiny and evaluation.

3.5.2 Construct validity

Construct validity measures the degree to which the researcher is measuring the right variables under study by means of the questions in the questionnaire, and/or whether the study is measuring what the researcher presumes (Oates, 2006). To enhance validity, a brief pilot survey was done with a subset of the target population (Laanti et al., 2011). The pilot questionnaire was distributed to 10 respondents, and feedback was incorporated into the questionnaire before embarking on a full-scale study.

3.5.3 Internal validity

This is a concerned with the cause effect relationship i.e. the question is asked as to whether the research is justified in claiming that A causes B, and that a causal or correlation relationship exists in reality (Oates, 2006). In other words, internal validity is concerned with whether the results are objective or free from bias, or whether there are rival explanations or extraneous variables. Unfortunately, given the nature of this study, some other variables might have been involved since this study was undertaken in a social context where there is some uniqueness involved in each context. The former is the first source of threats to validity. The second source of validity threats may have come from the online data collection strategy, given that there could be duplicate submissions or responses not coming from the intended target group. Unfortunately, it was not possible to entirely hold all factors constant or eliminate them completely, which is only possible in a clinical laboratory setting, which is a setting that could have introduced bias and inaccuracies into the results. However, all necessary precautions were followed so as to identify other factors that could distort the findings and the following mitigating/isolation measures were employed.

To mitigate or isolate variables from the first source of validity threats (rival explanation or extraneous variables), the study was designed to also obtain information about the following rival explanatory variables: the experience of the project teams; the level of senior management and user involvement; project team and organisation size; project size; relevant on-the-job training; and any quality standards or frameworks used.

To mitigate or isolate variables from the second source of validity threats (online submission strategy), submissions were manually checked for duplicates, since a few open-ended questions

were included in the survey. The survey collection software was configured to check for duplicate submissions. In addition, the target nature of the selection and domain specific knowledge made it unlikely for someone not in the target population to respond.

3.5.4 External validity

This is concern with whether generalisations can be made about the research findings to other contexts, areas or times, beyond the original context where the study was carried out. Organisations in South Africa and globally were surveyed. The fact that responses also came from a global audience means generalisations are likely to be made. Similar and related studies have also been carried out, but not exactly like this one, so it is likely that comparisons might also be possible to corroborate findings.

3.5.5 Reliability

Reliability addresses repeatability of the study i.e. whether the same results will be obtained if the study is repeated. This is a bit tricky to assess firstly, because of the dynamic nature of software development and secondly because respondents can change their views over time, or deliberately give an opposite view (Oates, 2006). Repeatability of results is a key assessment criterion in positivist research, presupposing or assuming that there is some objective truth out there that needs to be investigated and reported. Another aspect regarding reliability is the neutrality of the research instruments i.e. the questionnaire and the researcher inclusive. Despite the possible biases and problems with repeatability, the following measures were taken to increase reliability: the researcher did everything possible to make sure that questions formulated were neutral i.e. not influencing the respondents to respond in a particular way. Secondly, the researcher took appropriate measures to make sure that questions formulated were not ambiguous, with clear and precise instructions how to complete the questionnaire. Thirdly, the questionnaire used for this study has been included as an appendix to this report for scrutiny, evaluation and independent testing and verification. Finally, the researcher remained neutral and objective throughout the entire study.

3.5.6 Ethical considerations

This refers to norms for conduct that distinguish between acceptable and unacceptable behaviour. In research, it particularly has to do with the rights of research subjects, participants or respondents. The researcher was fully aware and understood that those involved in the research have the right not to participate, the right to withdraw at any stage of the research, the right to give informed consent, the right to anonymity and confidentiality. This study complies with the UNISA Research Ethics Policy (2007). Respondents were invited to voluntarily complete the

questionnaire. The data collected was treated as private, confidential and the identity of the respondents was withheld. The researcher did everything possible to respect and uphold the ethical principle of honesty, objectivity, integrity, carefulness, openness, confidentiality and respect for intellectual property.

3.6 Conclusion

This chapter explored the research design and methodology that was employed to obtain results. Methods used were explained and their choice justified. The chosen research paradigm was positivism. Explanatory research was chosen as the applicable purpose of this study. The fact that studies these days tend to find themselves on the quantitative/qualitative continuum was highlighted.

Following from the above, it was underscored that this study lies within that interactive continuum, but leans more towards the quantitative paradigm. Deduction was briefly explained as the mode of reasoning underlying this study. Dictated by the nature of the research question, the research strategy and design chosen was correlational, as well as causal comparative, using a survey with a structured questionnaire as a data collection method. The sampling method used was clustered sampling. The data collected was captured and coded prior to using descriptive and inferential statistics for analysis. Validity and reliability constraints, as well as mitigation measures, together with ethical consideration, were also addressed. The assumptions implicitly underpinning this study were also sporadically articulated.

CHAPTER 4: DATA ANALYSIS AND RESULTS

4.1 Introduction

The focus of this chapter is on data analysis and interpretation as per methodology described in Chapter three. The purpose is to ascertain, firstly, the impact of agile methodologies on selected software quality parameters when compared to traditional methodologies, and secondly, to establish whether there is a correlation between methodology type or use and software quality in general. The software application used for analysis was Statistical Package for the Social Sciences (SPSS). SPSS was used due to ease of use and credibility amongst statisticians (Abbas et al., 2010). The type of statistical analysis undertaken was descriptive and correlational. Descriptive statistics techniques are normally used to describe and compare variables numerically, while correlational analysis techniques are employed to find relationships between variables (Saunders et al., 2009).

This section comprises five sections. The first section is a brief articulation of the data collection method employed, and sampling frame surveyed. The second section is a brief discussion of the reliability of the research instrument used. The third section is an analysis of background and demographic information. In this section, only frequency analysis was done, since the data type is categorical. In the fourth section, correlational analysis was conducted so as to establish association between research variables. The last section is a discussion, and conclusion of research findings.

4.1.1 Data collection

The sample frame was IT practitioners comprising mainly, but not exclusively, project managers, business analysts, system analysts, architects, developers, product managers, software test analysts, consultants and business stakeholders, who have worked in both agile and traditional environments. The questionnaire was distributed to these professionals in South Africa as well as placed online via a link courtesy of SurveyMonkey to three online professional groups for a period of three months. The survey collated 127 responses, 21 were incomplete, and therefore discarded. The remaining 106 complete responses were used for analysis.

4.1.2 Scale reliability

Validity concerns were extensively covered in the previous chapter, but scale reliability will be briefly revisited here. Reliability is concerned with the repeatability of the study, i.e. whether the same result can be obtained if the study is repeated. It is also concerned with trustworthiness of the research instrument (Oates, 2006). For this study, an actual reliability test was conducted to

test the reliability of the questionnaire. The purpose of the test was to determine the extent to which questions measuring software quality can be said to be internally consistent i.e. measuring the same construct. Cronbach's alpha coefficient is the most popular reliability test in this regard, evident from the broader literature. It computes a value that advises the researcher of the degree of internal consistency of the measurement scale. The value typically lies between 1 and 0. The closer the value approaches 1, the higher the reliability, and the closer it approaches 0, the lower the reliability Mitchell (1996). Cronbach's Alpha coefficient of reliability was computed for all variables measuring software quality, and the results are shown in Table 12.

Reliability Statistics

Cronbach's Alpha	Number of items
0.807	15

Table 12 : Reliability statistics matrix.

A score greater than 0.5 indicates high reliability (Mitchell, 1996). In this case, the score is 0.807, indicating that variables used to measure software quality were indeed highly reliable.

4.2 Analysis of background and demographic information

This section is a comprehensive analysis of results of background and demographic information. The main graphical models used for analysis in this section are bar graphs. They are normally used to visually represent data and show comparisons between categorical data or to represent the number of observations in a given category.

4.2.1 Methodology type

This question (see question 1, section 7: questionnaire) was meant to gauge the current extent of agile and traditional methodology adoption in organisations that were surveyed. The results are shown in Figure 3.

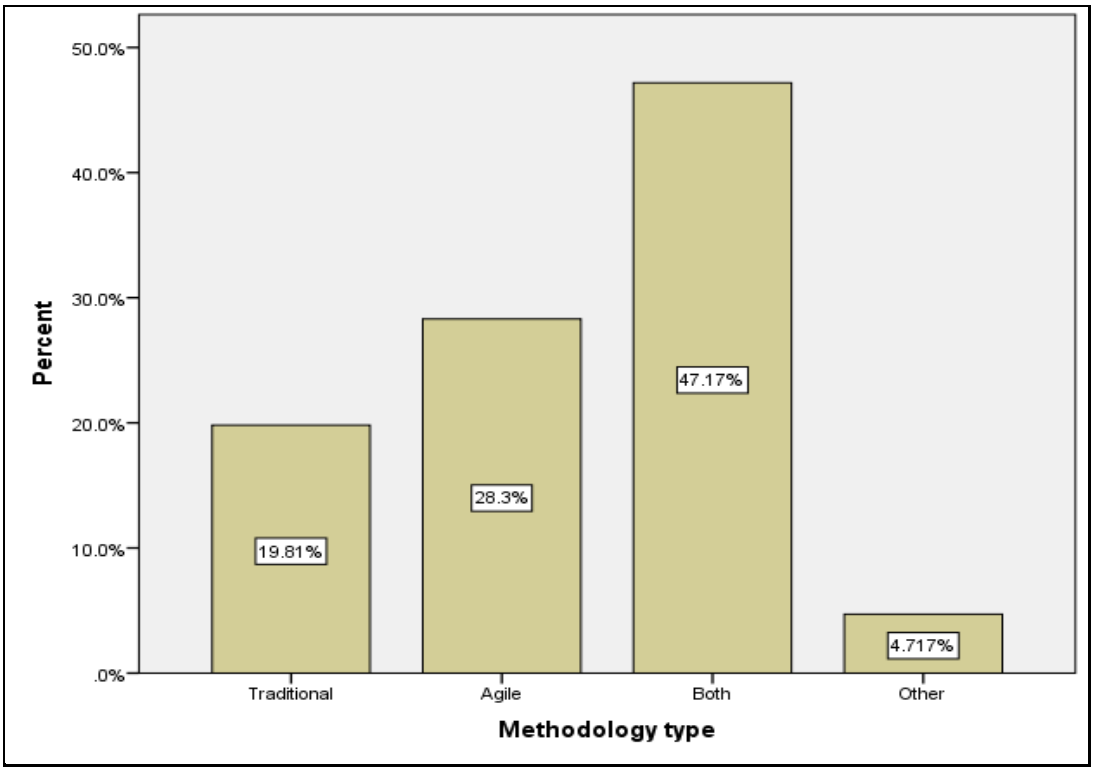

Figure 3 : Graphical representation of methodology adoption state.

The results show that 19.81% are using traditional methodologies, 28.3% are using agile methodologies, 47.17% are using both methodologies, and 4.75% are using other methodologies. Other methodologies mentioned by respondents were mainly methodologies developed in-house, or customised SDLC or Agile. These results corroborate previous findings that agile methodologies are becoming increasing popular, with the number of organisations embracing them increasing (Salo and Abrahamsson 2008; Zhang et al., 2010; Laanti et al., 2011).

However, despite the above-mentioned popularity of agile methodologies, in terms of increasing adoption, Rodriguez et al. (2012) have cautioned that most of the publications are generated by agile consultants, tool vendors, professional societies, and market research organisations; not from academics. This threatens the trustworthiness and validity of the research results, posing potential conflict of interest.

The results on agile popularity contradict previous studies that traditional methodologies are still the dominant methodologies (Zhang et al., 2010; Kendall et al., 2012; Moniruzzaman and Hossain, 2013). Furthermore, the results also show that most organisations are not locked into one methodological category, but are using features of each methodology to complement one another.

4.2.2 Definition of agility

The purpose of this question was threefold, namely: to find out what agile means to respondents; to find out if their responses concur with the core principles of the Agile Manifesto; and lastly, to ascertain whether a universal definition of Agility can emerge or be observed. The results are show in Figure 4.

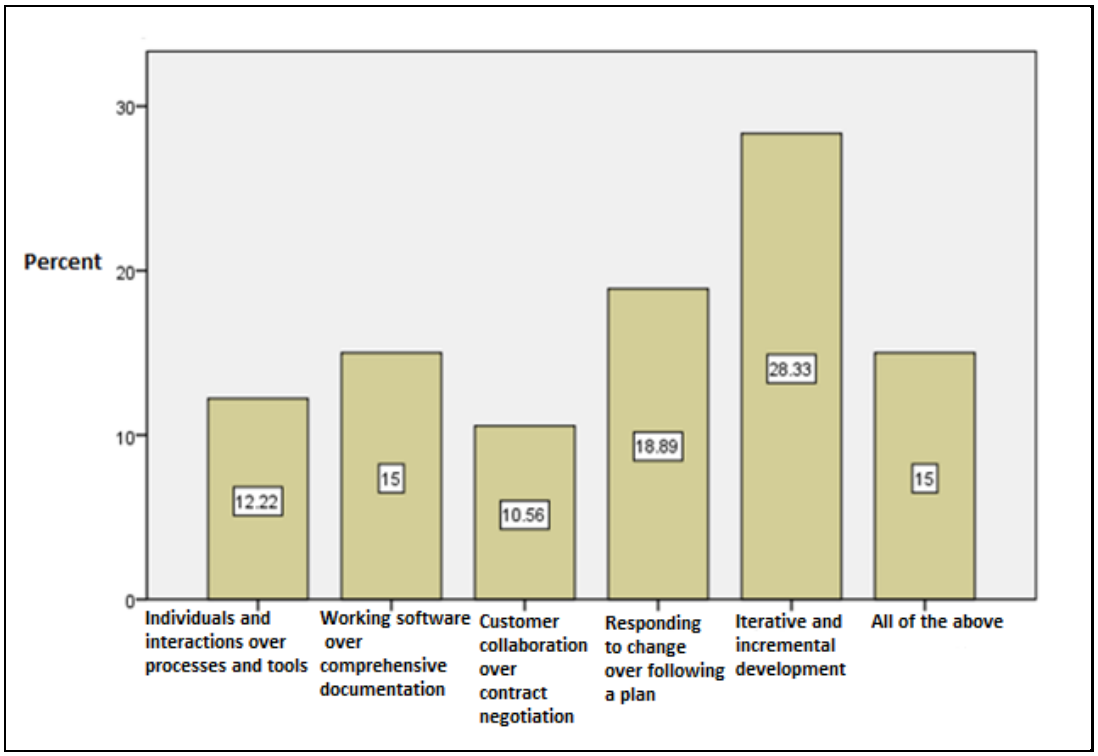

Figure 4 : Graphical representation for definition of agility.

Incorporated in the question was also an option for ‘other’. No respondent came up with another definition of agile. The reason for this might be because selecting ‘other’ would mean entering

free text which respondents might find to be more cognitively demanding as opposed to just conveniently selecting available options. Given this scenario, they were all restricted to select from the available options.

The results show that '*iterative and incremental development*' was selected by most respondents (28.33%) as the definition of agility. These results are consistent with the definition of agility in certain literature sources (Leau et al., 2012; Moniruzzaman and Hossain, 2013; Mohammad et al., 2013). However, once again, iterative and incremental development is not a feature unique only to agile methodologies. Other methodologies, including traditional ones also have iterative and incremental elements inherent to them (Larman and Basili, 2003; Rao et al., 2011). Jiang and Eberlein (2009) conducted a study seeking to determine the historical links between traditional and agile software development methodologies, concluding that many of the foundational practices and principles used in agile methodologies have historical links with traditional methodologies, and are therefore not entirely novel. It will therefore be technically inaccurate and misleading to base the definition of agility only or entirely on the iterative and incremental feature.

An implicit observation from the graph is that certain respondents only selected one option, others selected two, while others selected four, and only 15% selected all of the above. This means that 85% of respondents do not agree that agile means 'all of the above'; otherwise it would have been selected by all respondents. This situation highlights the need for common standards for agile methodologies, as well as the need for a consensus definition of agility. The results also imply that some practitioners are applying at least some of the core principles of the Agile Manifesto differently, as may be inapplicable or ill-suited to their unique situations, nonetheless concluding that their methodologies are agile. Several studies have vehemently argued before that a universally accepted definition of agility is not readily observable, or is non-existent (Conboy and Fitzgerald; 2004; Aitken and Ilango 2013; Hummel 2014). This study has corroborated those findings.

4.2.3 Agile methodology type

The purpose of this question was to identify current adoption trends of agile methodologies and also to establish whether respondents were familiar with the agile methodology they chose to practice. The results are shown in Figure 5.

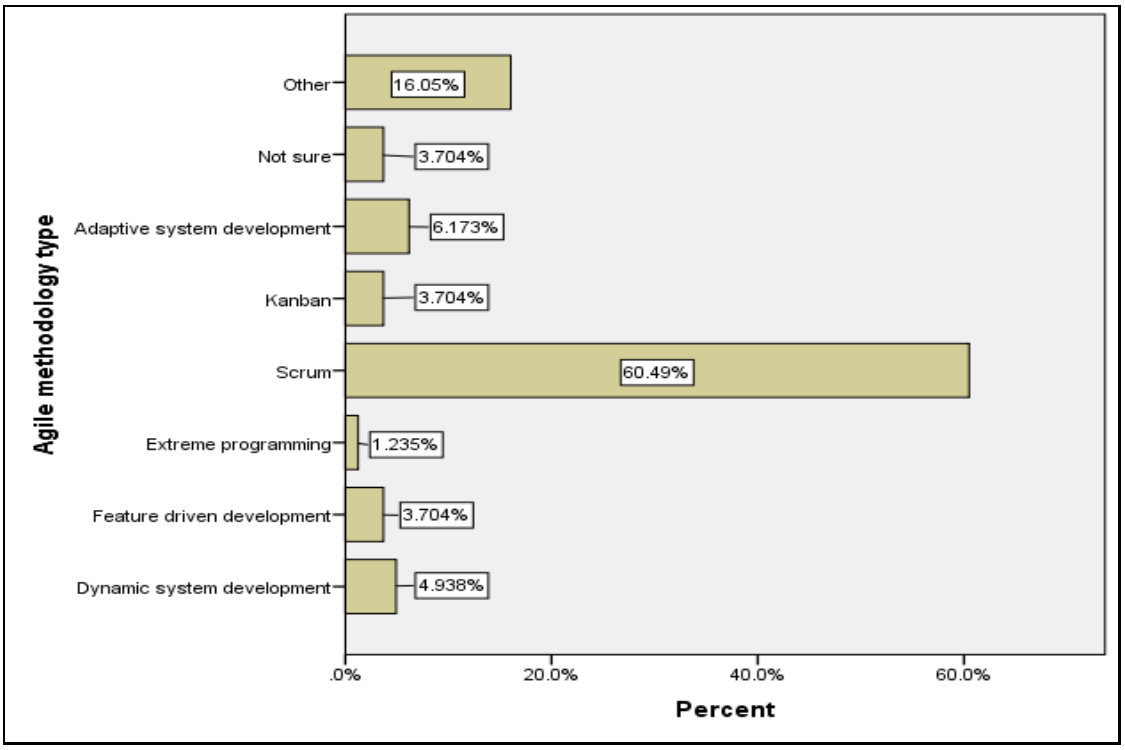

Figure 5 : Graphical representation of agile methodology type.

It is clearly evident from the results that Scrum is the most widely used agile methodology, with more than half of organisations surveyed practicing it. This finding is consistent with previous research (Conboy 2009; Rao et al., 2011; Moniruzzaman and Hossain, 2013). Surprisingly, and contrary to previous reports, the much talked about XP got the lowest score of 1.25 percent. This is a striking contradiction of previous studies that found XP to be the main pillar and most popular or representative of the agile methodologies (Zhang et al., 2010; Ahmed et al., 2010; Kendall, 2012). The under-representation of XP, contrary to expectations, might well be the beginning of a new trend. Kapitsaki and Christou (2014) found that Scrum is fast gaining ground when compared to other methodologies, especially XP. Meyer (2014) asserted that the assertive nature of XP, compelling the use of techniques as obligations rather than as options, has hampered its overall adoption. It is also interesting to note that the second mostly widely used agile methodology was noted to be 'other' than the options presented. Such methodologies were mainly reported by participants to be a combination of agile methodologies, or customised/in-house agile methods. Another observation of worth noting is that 3.7% of respondents were not certain as to which agile methodology they were practicing.

This scenario again underscores the need for research-based definitions of agility, as argued by Conboy and Fitzgerald (2004). Finally, it was observed that no respondent (0%) selected Lean

development, and Crystal agile methodologies, even though the literature asserted that they are also popular (Rao et al., 2011; Moniruzzaman and Hossain, 2013).

4.2.4 Duration of agile practice

This question aimed to determine maturity levels of organisations practicing agile methods in terms of the number of years since establishment. The results are shown in Figure 6.

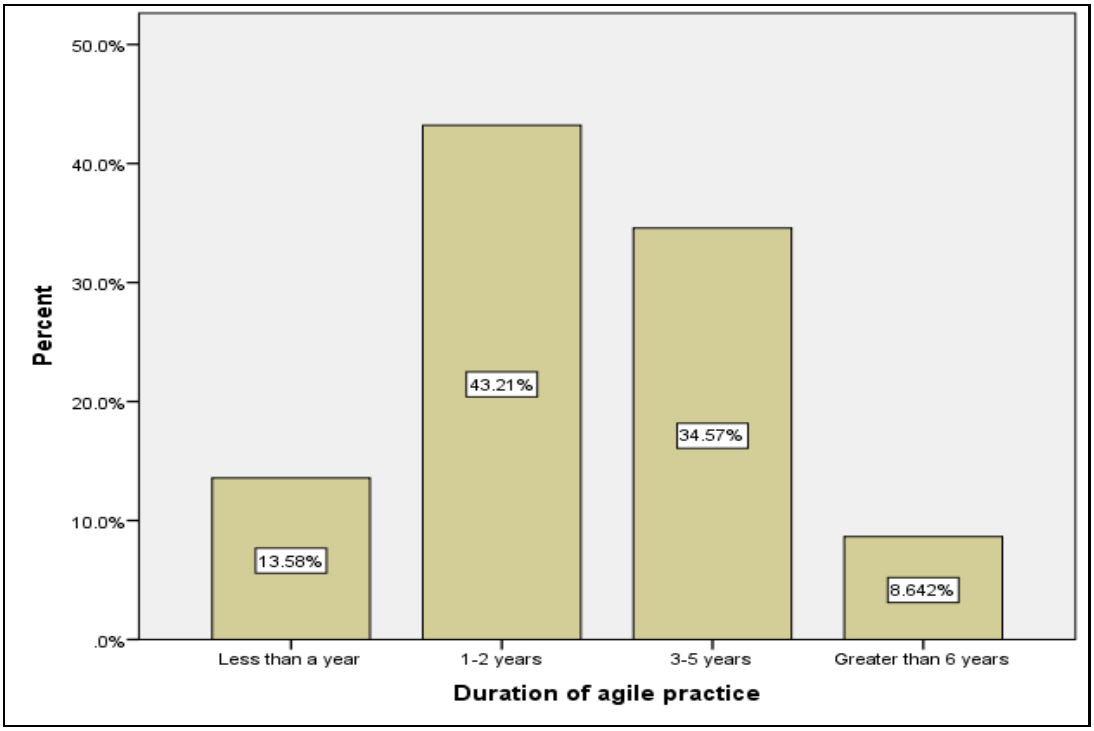

Figure 6 : Graphical representation for duration of agile practice.

The results show that 13.58% of organisations surveyed are below one year in terms of agile experience. This indicates that agile methodologies are increasingly gaining popularity with more organisations adopting them. The results also show that a greater proportion of organisations (43.21%) have an agile experience of between 1-2 years. The interpretation is that most organisations are still at an infant stage with regards to agile experience. About a third of organisations (34.57%) surveyed have an experience of 3-5 years, indicating mid-level experience. The results also show that a relatively small percentage (8.642%) of organisations surveyed have really gained enough agile experience of six years and beyond. This is understandable; given that agile is still a relatively new software development paradigm.

4.2.5 Experience of the agile team

Similar to the previous question for organisations (see section 4.2.4); this question was meant to gauge the experience levels of agile teams of respondents. The results are shown in Figure 7.

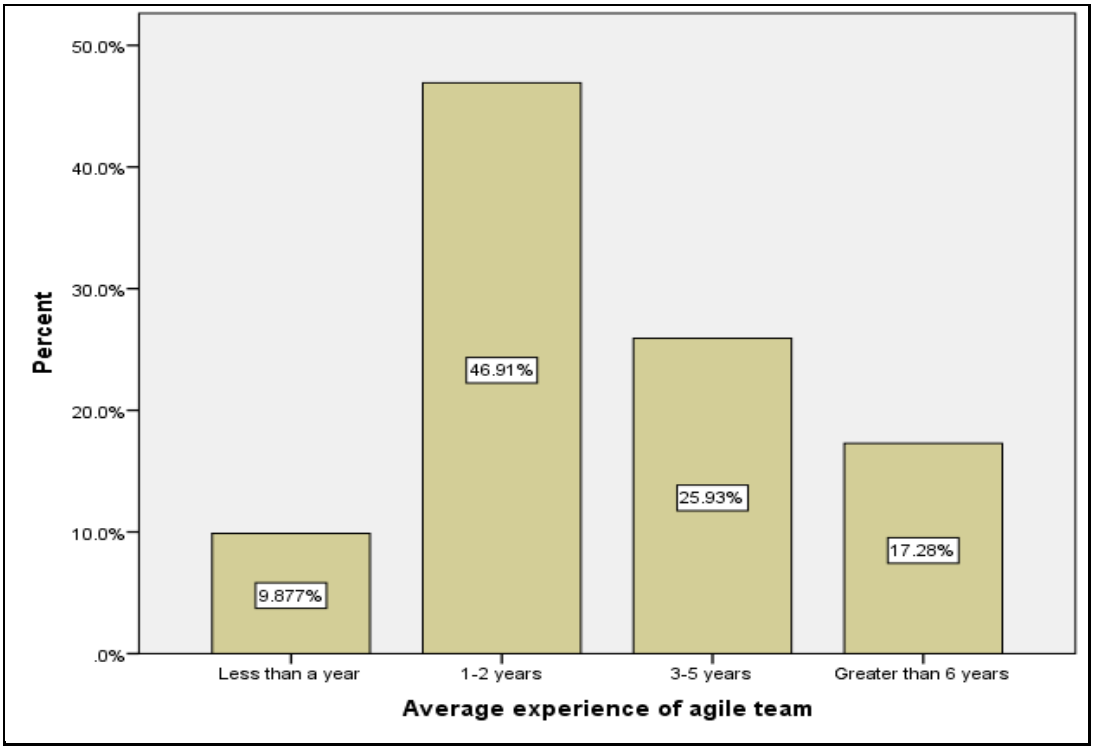

Figure 7 : Graphical representation of agile team experience.

The results are similar to previous results for agile organisational experience (see section 4.2.4). This is another proof that the scale or instrument used is internally consistent. They show that more teams are beginning to use agile, with most teams still at a novice level of 1-2 years. A quarter (25.93%) of the teams has mid-level experience, and 17.25% have an experience greater than six years and beyond.

4.2.6 Size of project team

The objective of this question was to obtain information about the size of project teams. The results are shown in Figure 8, arranged by methodology type.

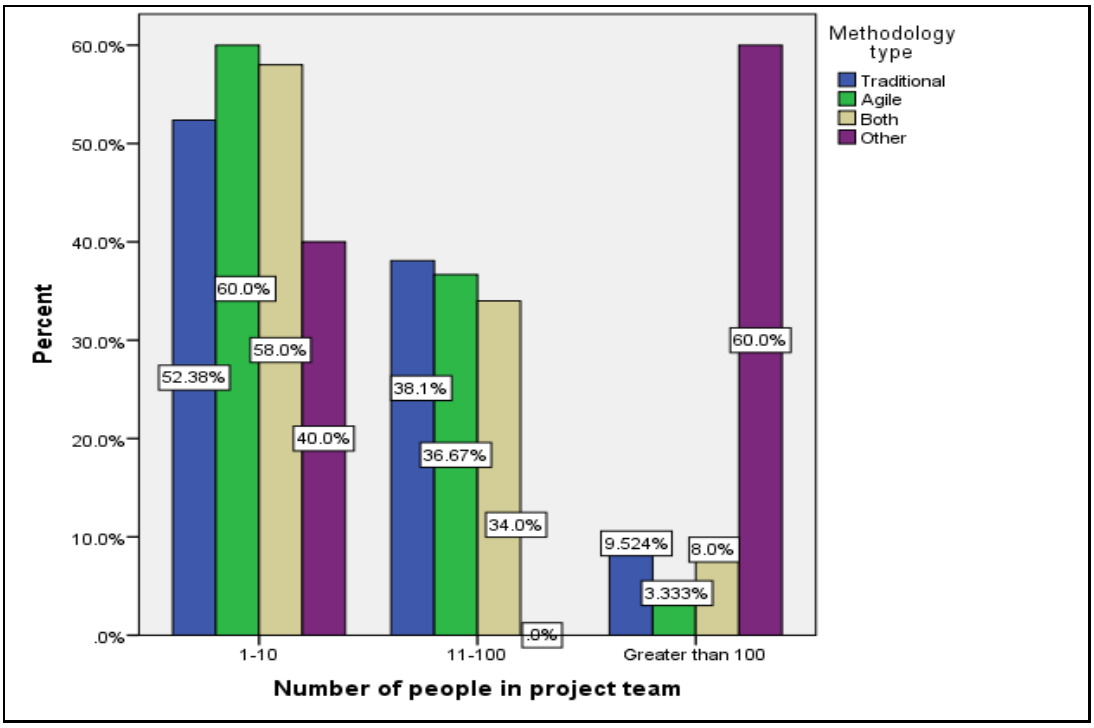

Figure 8 : Graphical representation of project team size clustered by methodology type.

The results show that (60%) of those who selected an agile methodology had small projects team sizes (i.e. 1-10), compared to traditional and both methodologies. They also show that agile is less prominent (only 3.3%) when the team size is greater than 100. This observation is consistent with previous findings, and suggests that agile methodologies are suitable for small teams. Several studies have reported that agile methods work well in small environments or projects (Dyba and Dingsoyr, 2008a; Barlow et al., 2011; Saxena and Kaushik, 2013), but empirical evidence of their success in general, and in large environments in particular, remains elusive (Abrahamsson et al., 2010; Barlow et al., 2011).

It is also interesting to note that 'other' methodologies are more prominent, (60%) when the team gets larger, i.e. greater than 100. Other methodologies from respondents were mainly customised agile or SDLC, as well as methodologies developed in-house. This therefore suggests that other methodologies are suitable when the project increases in size.

4.2.7 Average size of projects

The objective of this question was to obtain information about the average size of projects. The results are shown in Figure 9, arranged by methodology type.

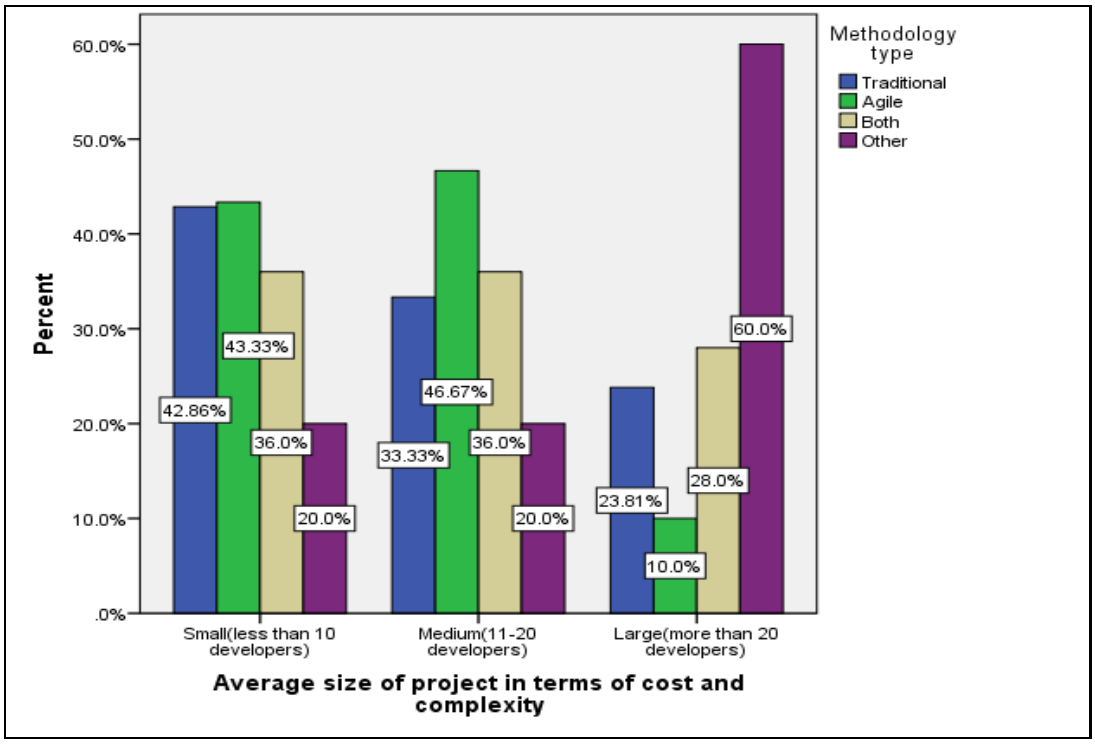

Figure 9 : Graphical representation of project size clustered by methodology type.

The results further support the findings earlier (see Section 4.2.6) that agile methodologies are suitable in small environments. It is evident from the graph that agile is relatively less prominent (only 10%) when the project becomes larger or more complex, and more prominent (46.6% and 43.3%) when the project size gets smaller or becomes less complex. Another finding supported by the bar graph is that 'other' methodologies (60%) are suitable when the project increases in size or complexity. As alluded to earlier, other methodologies mentioned by respondents were mainly methodologies developed in-house, or customised SDLC or agile. The interpretation here is that the prescriptive nature of methodologies makes them less likely to be applicable in their entirety when the project size becomes increasingly large or complex, and therefore need to be customised or adapted to be applicable in those environments.

4.2.8 Size of organisation

The aim of this question was to obtain information about organisational size. The results are shown in Figure 10, clustered by methodology type.

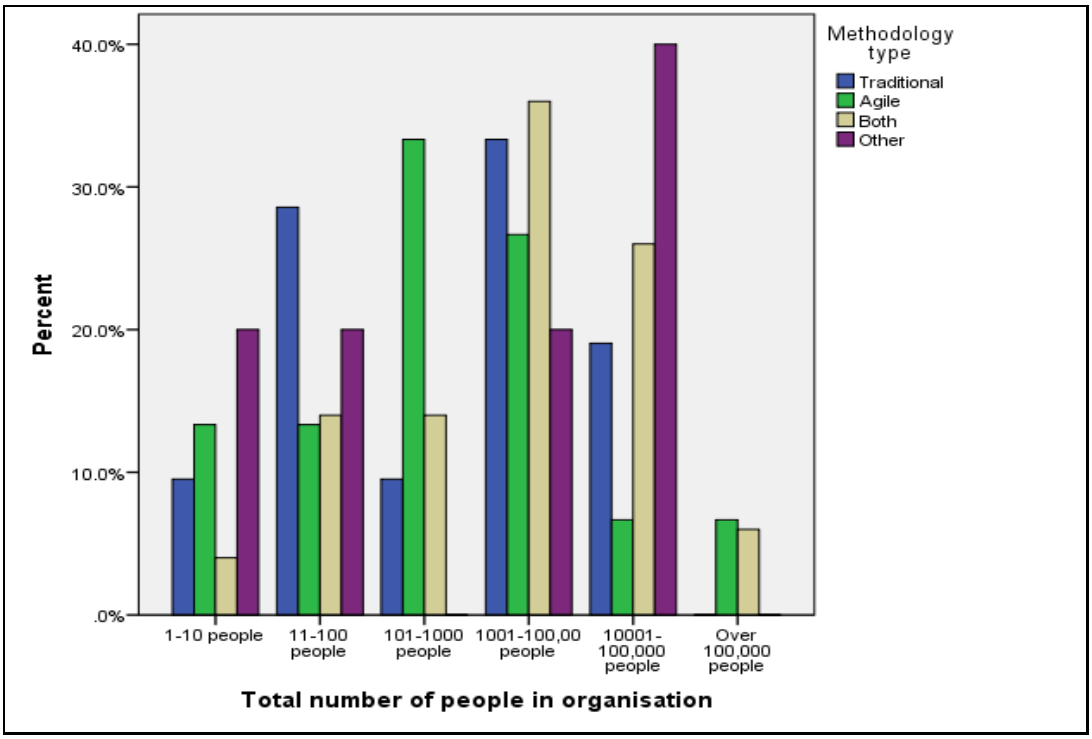

Figure 10 : Graphical representation of organisational size clustered by methodology type.

The first observation is that organisations of all sizes participated in the survey from small, medium to large. The second observation is that agile is prominent when the organisational size is small, compared to traditional methodologies. The results further show that 'other' methodologies are suitable when organisational size gets larger, thus corroborating findings in the previous paragraphs (see section 4.2.6 and 4.2.7).

It is therefore clear that agile methodologies are suitable for small environments, and need to be adapted when the project team, size and organisation becomes increasingly large or complex.

4.2.9 Quality standards

The purpose of this question was twofold: firstly, as a contingency to control for an extraneous variable in case a significant relation is found between methodology type and a given software quality parameter; and secondly, to check whether organisations are employing any software quality standards in their system development processes. The results are shown in Figure 11.

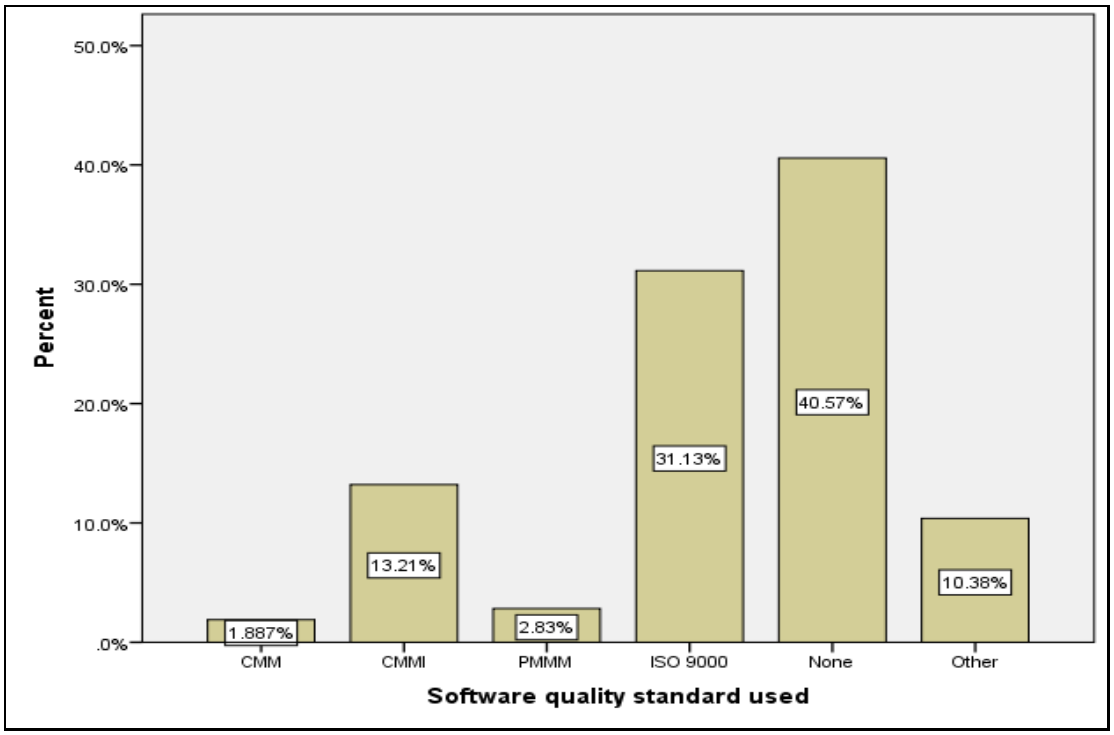

Figure 11 : Graphical representation of quality standards used.

It is clearly evident from the results that ISO 9000 is the most widely used quality standard followed by CMMI. It is worth noting that 40.5% of organisations surveyed do not follow any quality standards. 'Other' quality standards mentioned by respondents were mostly in-house standards, or a combination of standards.

Furthermore, a positive but weak relation was found between Software quality standard and ease of system testing (p=0.017). This will be explained further in section 4.3.2 (analysis of variables). No relationship was found between quality standard used and other selected software quality parameters.

Further to the forgone analysis, so as to ascertain how and which quality standards are being used when different methodologies are employed, a bar graph of quality standards used clustered by methodology type was generated, with the results shown in Figure 12.

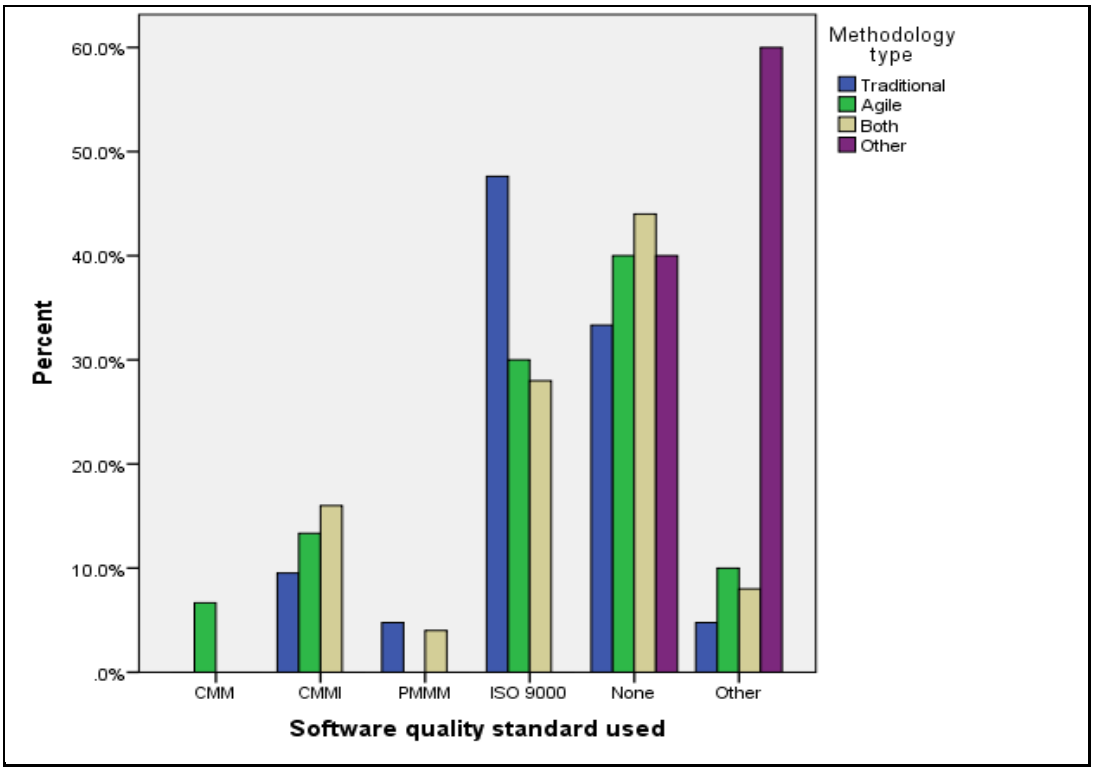

Figure 12 : Graphical representation of quality standard used clustered by methodology type.

The results show that only CMM is being used by agile methodology users, but that agile is absent with respect to PMMM. ISO 9000 is employed somewhat more by traditional methodology users. Compared to traditional methodologies, agile methodologies were more prominent (40%) when it came to not using any software quality standard. The researcher speculates that agile enjoys a higher degree of representation compared to traditional methodologies, when it comes to not using quality standards, because their proponents claim that agile methodologies have built-in quality abilities (Huo et al., 2004; Bhasin, 2012). But again, the graph also implicitly shows that 60% of agile users are employing software quality standards. This again underscores the need for common standards for agile, and raises questions about reports linking agile to improved software quality (Huo et al., 2004; Ahmed et al., 2010; Bhasin 2012). One would then wonder if the software quality was improved as a consequence of using purely agile (without any external quality standard), or as a result of software quality standards used. It is a contentious issue, and warrants further investigation.

4.2.10 Stakeholder participation on projects

The purpose of this question was twofold: firstly, as a contingency to control for a rival explanation variable (extraneous) in the case of a significant relationship between methodology type and a given software quality parameter; and secondly, to assess the level of stakeholder participation on projects. The results are shown in Figure 13.

Figure 13 : Graphical representation of stakeholder participation on projects.

The results show that approximately 75% (34.9% + 40.5%) of respondents agreed that stakeholders actively participated on projects. Seventeen percent (17%) were neutral, and a small proportion (6%) of respondents disagreed.

Furthermore, a positive but weak significant relationship was found between active stakeholder participation and 'ease of system interactivity (p=0.047). This will be explained further in section 4.3.10 (analysis of research variables). No relationship was found between active stakeholder participation and other selected software quality parameters.

4.2.11 Attendance of mandatory training and workshops

The goal of this question was twofold; as a contingency to control for an extraneous variable in case a significant relationship between methodology type and a given software quality parameter; and secondly, to assess training and workshop attendance levels of project stakeholders. The results are shown in Figure 14.

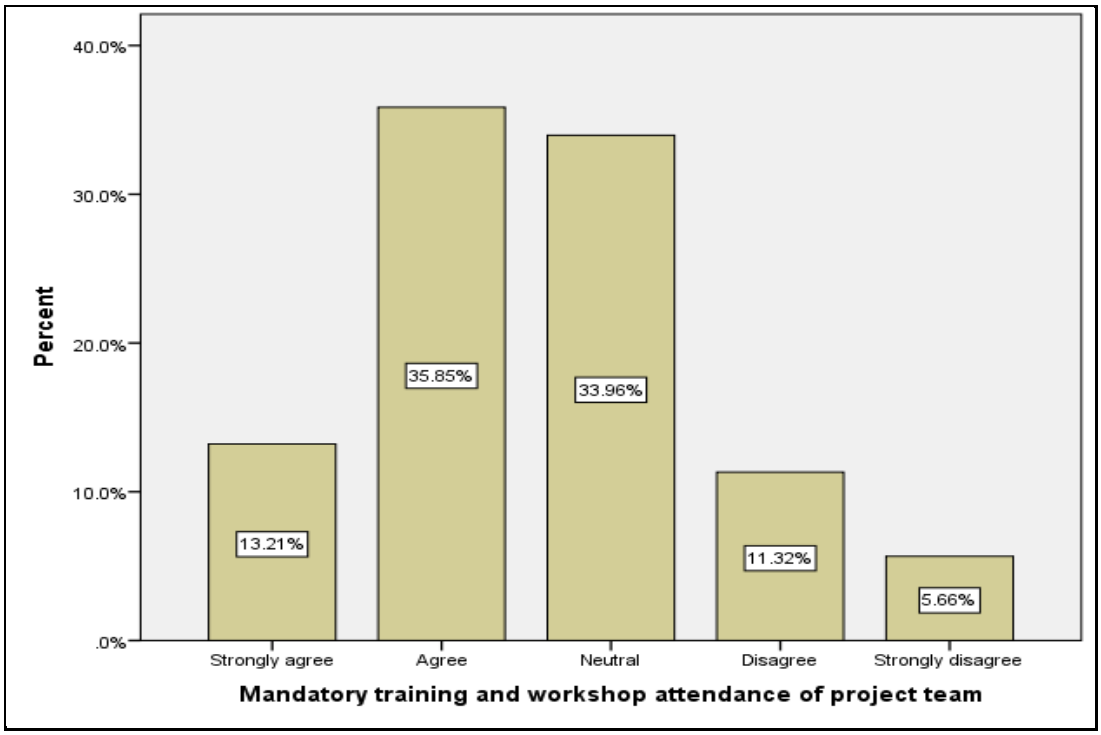

Figure 14 : Graphical representation of mandatory workshop attendance and training.

The results show that approximately 49% (13% + 36%) of respondents attended workshops and training. Approximately 34% were neutral, and 17% (11% + 6%) did not attend workshops or training.

Furthermore, a positive but weak significant relationship was found between active workshop attendance/training and ease of system navigation (p=0.031). This will be explained further in section 4.3.9 (analysis of research variables). No relationship was found between workshop attendance/training and other selected software quality parameters.

4.2.12 Respondents position within organisation

The purpose of this question was to ascertain whether or not respondents had the necessary domain knowledge to complete the survey in a way that will produce meaningful results. The results are shown in the Figure 15. The bar graph shows that a greater percentage of respondents were business/system analyst, followed by project managers, where test analyst was third, in terms of representation. The representation of all relevant portfolios as depicted in the graph (Figure 15) is therefore ample evidence that the sample respondents had the relevant domain knowledge, and also understood what software methodologies and software quality entails.

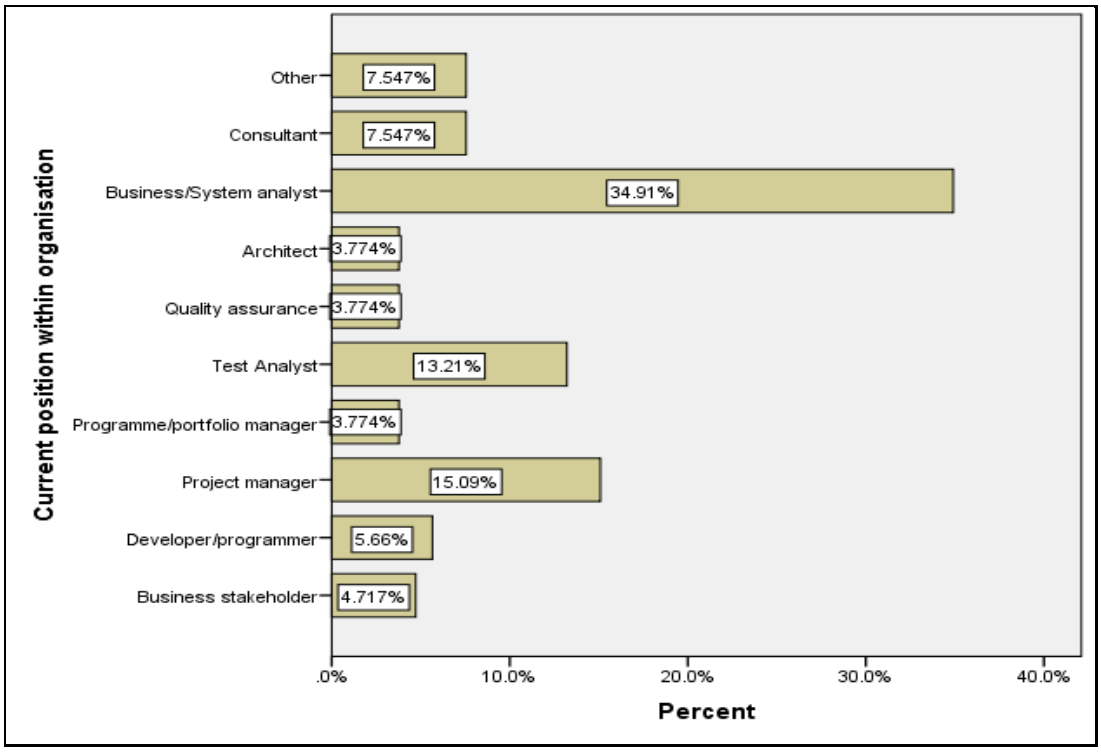

Figure 15 : Graphical representation of organisational position of respondents.

4.2.13 Identity of respondent's organisation

The goal of this question was to establish organisational representation of respondents. Banking and insurance was the most prominent, next was computer-related, and third was telecommunications. The bar graph (Figure 16) also shows that most organisational types were represented.

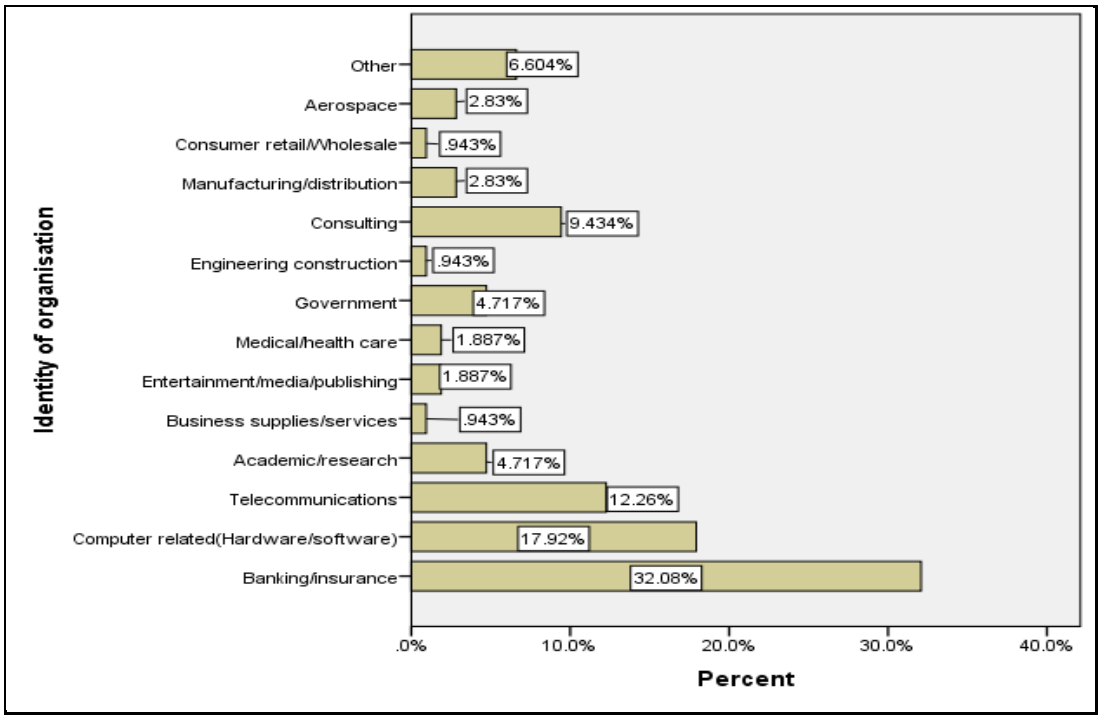

Figure 16 : Graphical representation of organisational identity of respondents.

4.2.14 Respondents country of residence

Finally, the rationale behind this question was to gauge global representation of respondents. Out of the 106 participants surveyed, 72 were from South Africa, seven from India, six each from UK and USA, three from Australia, two from Canada, and one each from Spain, Germany, New Zealand, Brazil, Ethiopia, Mexico, Serbia, Belgium, Nigeria and Sweden.

The interpretation is that it was a global study that represented all major continents of the world, namely Africa, North America, South America, Europe, Asia and Australia/New Zealand.

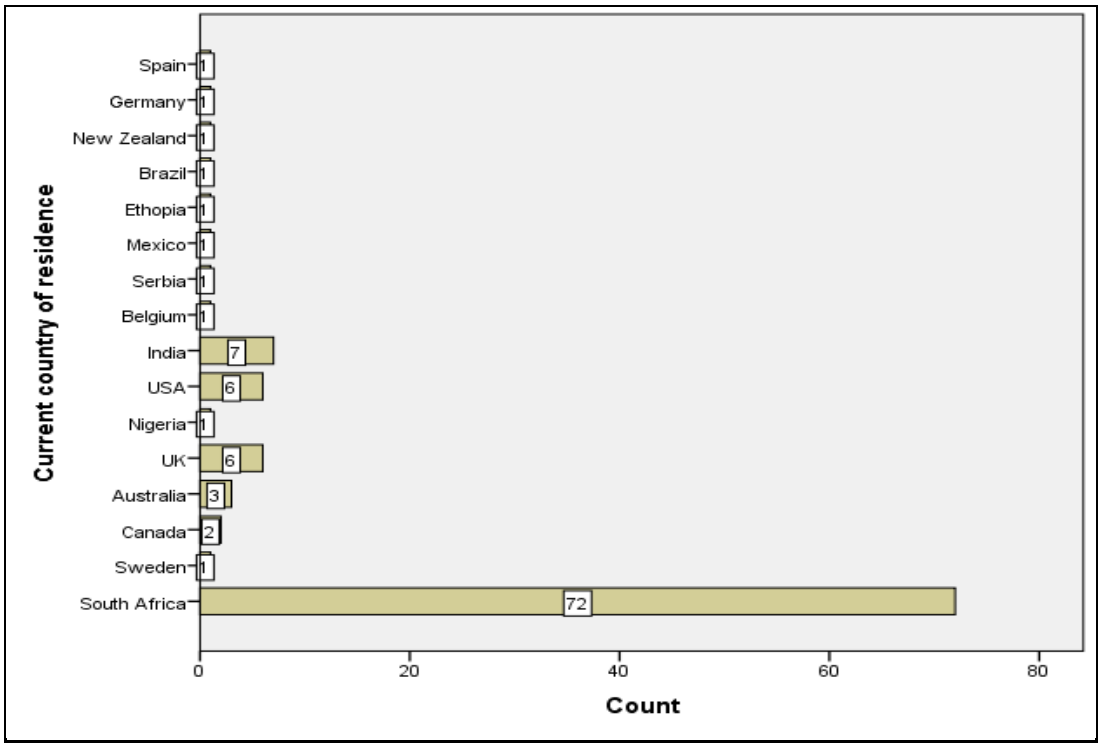

Figure 17 : Graphical representation of participant's country of residence.

4.3 Analysis of research variables

This section is a comprehensive and critical analysis of research variables identified for this study. The overarching objective of this research was to ascertain how selected software quality parameters are impacted if agile methodologies are used, as compared to traditional methodologies. In doing this, the study did also indirectly or implicitly establish whether or not certain software quality parameters can be linked to a system development methodology.

The main graphical models used for analysis in this section are bar graphs. Statistical test of significance were also employed to ascertain correlations between variables.

4.3.1 Ease of system maintenance

The purpose of this question was to establish the effect on *system maintenance* when agile methodologies are used, as compared to traditional methodologies. The question made use of the distinction in Table 13.

Systems	Traditional system	Agile system
Project characteristics: **Documentation**	Enormous and detailed documentation produced.	Light and documentation produced minimal.

Table 13 : Documentation distinction matrix: Ease of system maintenance.

The distinguishing characteristic is documentation extent. The results are shown in Figure 18.

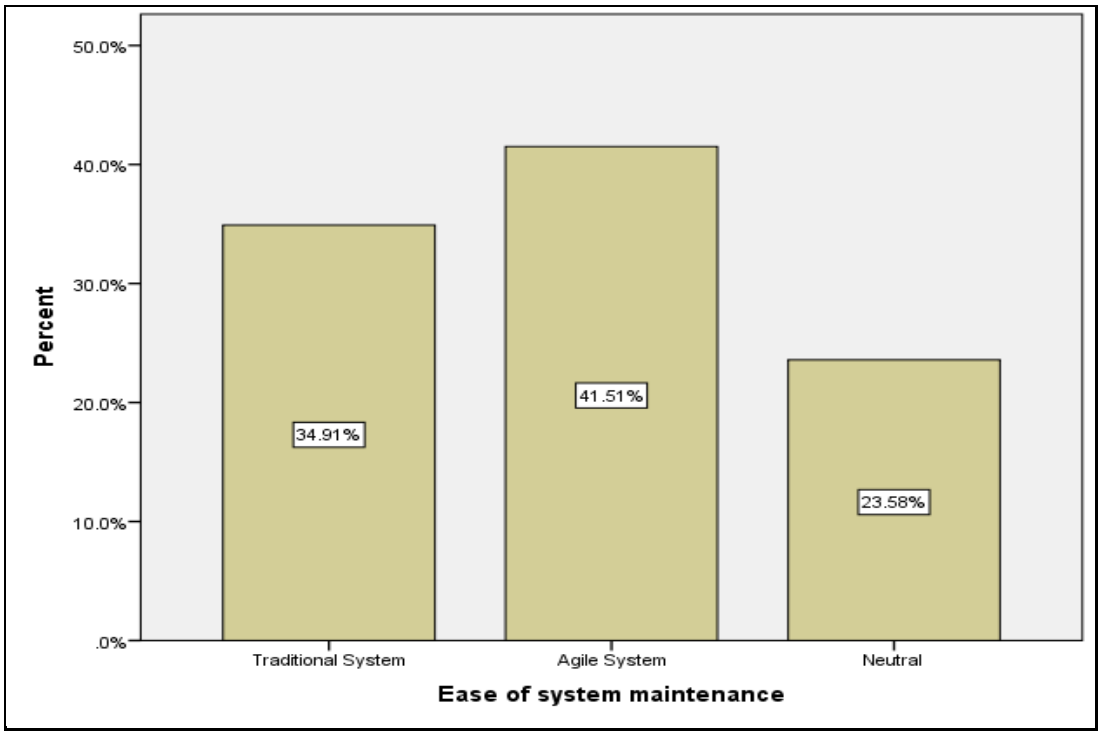

Figure 18 : Graphical representation of ease of system maintenance.

The results show that more respondents (41.51%) agreed that a system is likely to be easier to maintain when there is minimal documentation, as compared to when there is enormous documentation. In other words, maintenance will be easier if an agile methodology is used, as compared to when a traditional methodology is employed. Traditionalists advocate enormous documentation, while agile proponents emphasise minimal documentation. The puzzle here is that the definition of enormous or minimal documentation is not precise. However, it is logical and authors agree that a certain minimum amount of documentation is indeed needed for system maintenance (Pressman, 2010; Ahmed et al, 2010; Amir et al., 2013); but the question of what is enough documentation or to what extent documentation ought to be produced is not an exact science. In an attempt to explain the above graph, one might speculate that when there is enormous or detailed documentation, the support personnel or engineer will become overloaded with information, thus complicating the maintenance process. On the contrary, if there is minimal documentation, it becomes almost effortless to peruse the documentation and readily maintain the system.

However, against the above-mentioned backdrop, to be able to identify any apparent link between methodology use and ease of system maintenance, the point of departure is the fundamental assumption that there is no relation between them. This assumption is called the null hypothesis. This hypothesis postulates that there is no genuine relationship between any

variables, until proven (Oates, 2006). For a relationship to be uncovered, statistical test of significance needs to be computed for the data collected. Statistical tests of significance are normally used to compute a value that will indicate to the researcher whether the observed relation occurred by chance, or whether it indeed exists. This value is derived by computing the probability *(p)* of a relationship as being a result of chance (Oates 2006). If the probability of a relationship determined by chance is computed to be greater than 0.05 (one in 20), the null hypothesis is upheld i.e. there is no relationship between variables. On the contrary, if the probability of the relationship being down to chance is computed to be less than 0.05 (one in 20), the association is regarded as statistically significant i.e. there is a relationship between variables (Oates 2006). Several test of significance exist but for this study, Fisher's exact test was used to compute the *p* values. As perfectly applicable with this study, Fisher's test is normally employed in the following cases: to show interdependencies between variables in a contingency table, when the dependent and independent variable are both categorical; when the samples size is small (less than 1000); and when the Chi square test of independence assumption of count per cell is being violated (Everitt, 1992; Agresti, 2007; McDonald, 2009).Furthermore, Fisher's test was used because it is more accurate than the Chi square test of independence when the sample size is small (McDonald, 2009). Contingency tables are normally used to find correlations between categorical or nominal data (Olivier, 2009).

Therefore, to test the link between agile methodology use and ease of system maintenance, Fisher's exact test was conducted and a p value of 0.97 was obtained. Normally, the researcher will look for $p < 0.05$ to conclude that a statistically significant relationship exists, but in this case, the *p* value of 0.97 is clearly greater than 0.05. So the null hypothesis is upheld. The Conclusion is that there is no relationship between agile methodology use and ease of system maintenance, and secondly, that ease of system maintenance is not linked to any system development methodology. This also explains why 23.5% of respondents remained neutral.

4.3.2 Ease of system testing

This question was meant to establish the effect on *system testing* when agile methodologies are used, when compared to traditional methodologies. The contrasting project features are shown in table 14.

Systems	Traditional system	Agile system
Project characteristics : **Documentation**	Enormous and detailed documentation produced.	Light and minimal documentation produced.

Table 14 : Documentation distinction matrix: Ease of system testing.

The differentiating characteristic is documentation extent. The results are shown in Figure 19.

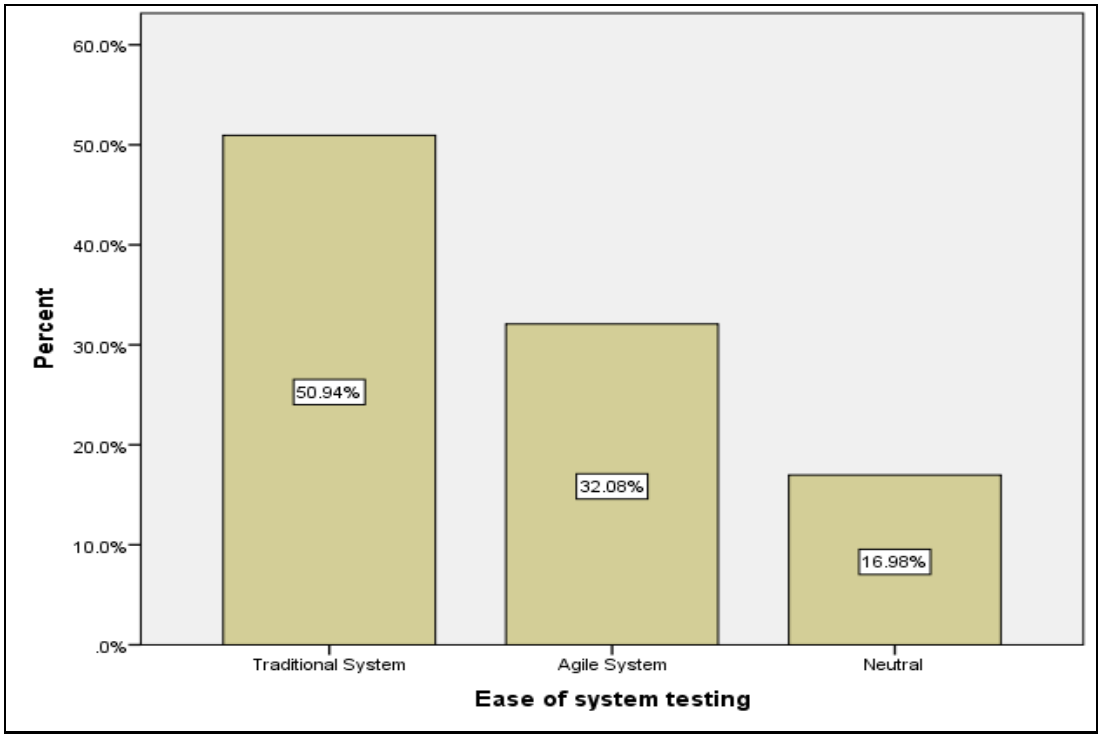

Figure 19 : Graphical representation of ease of system testing.

The results show that more respondents (50.94%) agreed that a system will be relatively easier to test so as to uncover defects when there is enormous documentation. In other words, system testing will be easier if a traditional methodology is used, compared to when an agile methodology is employed. However, it is logical, and authors agree that documentation is needed as a communication tool amongst project stakeholders (Pressman, 2010; Amir et al., 2013). Documentation in this case is used as input to derive test cases, and also as a reference to validate whether system functionality is consistent, as specified in the earlier phases of the project. The elusive question is what extent of documentation is required to ensure that testing is made easier.

To test the link between methodology use and ease of system testing, Fisher's exact test was conducted, and a *p* value of 0.014 was returned. The *p* value in this case is less than 0.05 and indicates a significant relationship between traditional methodology use and ease of system testing. However, statistical test of significance, like Fisher's test, do not show the strength of the relationship. To test the strength of the relationship, Cramer's V was used as opposed to Phi coefficient. Phi is normally used when the contingency table design is 2 x 2, or when both variables are dichotomous (Saunders et al., 2009). An example of this kind of design is the

relationship between gender (male and female) and smoking (Yes or No). Both gender and smoking have two categories. In cases where the table design is not 2 x 2, Cramer's V is used to ascertain the relationship strength (Saunders et al., 2009). This study has a 4 x 3 design i.e. (traditional, agile, both, and other) x (traditional system, agile system and neutral). Cramer's V value typically lies between 0 and 1. The relationship is stronger when the value approaches 1 and weaker when it approaches 0 (Saunders et al., 2009). The corresponding Cramer's V value in this case was 0.3, indicating weak relationship strength between traditional methodology use and ease of system testing.

In addition to the above, and as referred to earlier in section *4.2.9*, a positive but weak relationship was also found between software quality standards used, and ease of system testing in cases where traditional methodologies were selected (p=0.017). Therefore, in this particular scenario, a meaningful conclusion becomes a daunting task, because the observed effect may have been due to an extraneous variable (software quality standard), rather than to the use of traditional methodology. Further investigations are needed to draw a conclusion.

4.3.3 Ease of system training

This question sought to establish the effect on *system training* when agile methodologies are used, compared against traditional methodologies. The differences are shown in table 15.

Systems	Traditional system	Agile system
Project characteristics: **Documentation**	Enormous and detailed documentation produced.	Light and minimal documentation produced.

Table 15 : Documentation distinction matrix: Ease of system training.

The differentiating criterion is documentation extent. The results are shown in Figure 20.

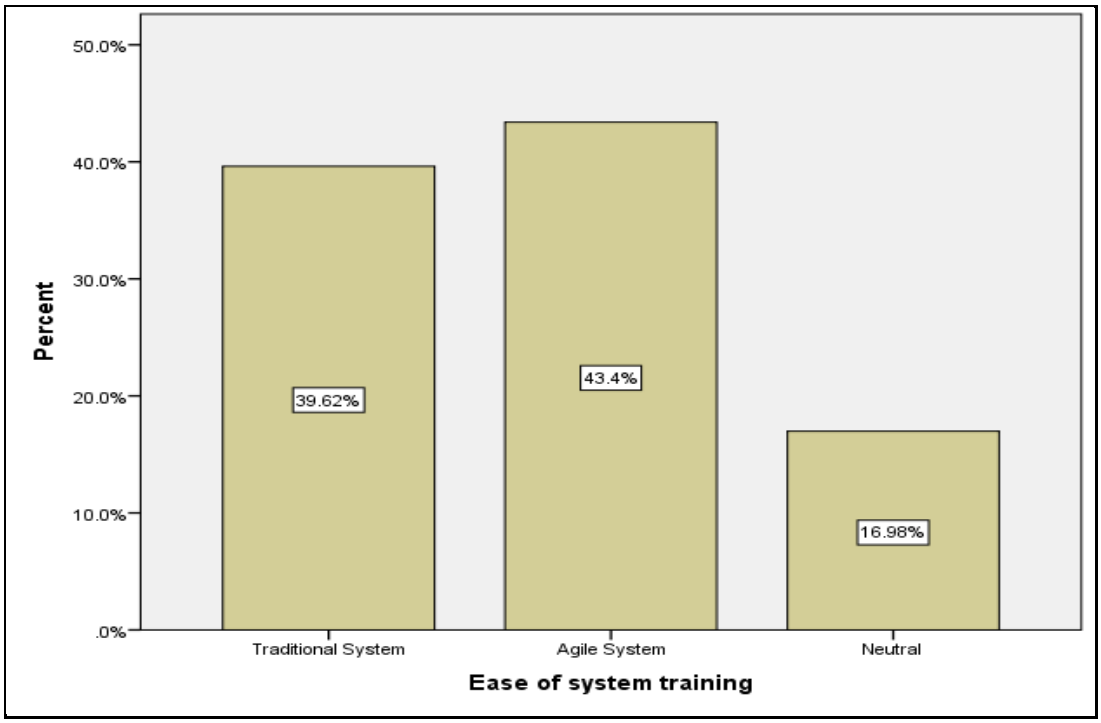

Figure 20 : Graphical representation of ease of system training.

The results show that more respondents (43.4%) agreed that it would be somewhat easier to train system users when there is minimal documentation. In other words, system training would be easier if an agile methodology is used, compared to when a traditional methodology is employed. Documentation in a software development process is normally used as a communication tool amongst stakeholders. In this case, it is used as input to develop training material as well as user manuals. It is a question of determining the amount of documentation necessary to ensure training is easier. Respondents of this question leaned more towards the agile approach of minimal documentation. A possible reason for this observation might be that vast and detailed documentation has the potential to overwhelm the personnel compiling the training material, thus complicating the system training process. On the contrary, when there is minimal documentation, it becomes easier for training material to be developed.

However, to test the link between methodology use and ease of system training, Fisher's exact test was conducted and a *p* value of 0.7 was obtained. The *p* value in this case is far greater than 0.05. The null hypothesis is therefore upheld. The conclusion is that there is no relationship between agile methodology use and ease of system training, and secondly, that ease of system training is not linked to any system development methodology.

4.3.4 Ease of system learning

This question sought to establish the effect on *system learning* when agile methodologies are used compared to traditional methodologies. The differences are shown in table 16.

Systems	Traditional system	Agile system
Project characteristics : **Documentation**	Enormous and detailed documentation produced.	Light and minimal documentation produced.

Table 16 : Documentation distinction Matrix: Ease of system learning.

The differentiating feature is documentation extent. The results are shown in Figure 21.

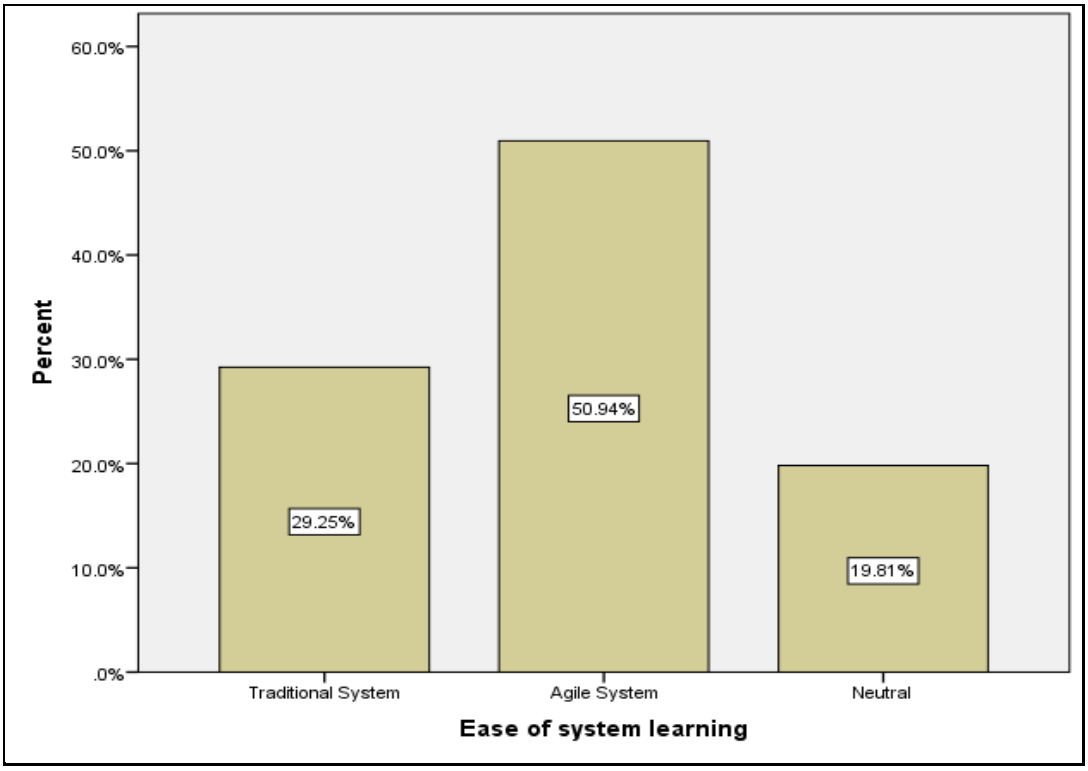

Figure 21 : Graphical representation of ease of system learning.

The results show that more than half of respondents (50.94%) agreed that it will be relatively easier to learn a system when there is minimal documentation. In other words, system learning will be easier if an agile methodology is used, compared to when a traditional methodology is employed. Documentation in this case refers to hard copy user manuals, as well as electronic versions embedded with the application. As was the case with training discussed earlier (see section 4.3.3), one can only speculate here that vast and detailed documentation has the potential to overwhelm system learners, thus complicating the learning process, or making the learning curve more steep. On the contrary, when there is minimal documentation, it becomes easier to efficiently peruse and learn the system. This trend of associating ease of system training

and ease of system learning with minimal documentation does not appear to be coincidental, and warrants further investigation.

However, to test the link between methodology use and ease of system learning, Fisher's exact test was conducted and a *p* value of 0.25 was obtained. The *p* value in this case is greater than 0.05. The null hypothesis is therefore upheld. The conclusion is that there is no relationship between agile methodology use and ease of system learning, and secondly, that ease of system learning is not linked to any system development methodology.

4.3.5 Robustness of system architecture

This question was meant to establish the effect or impact on *robustness of system architecture* when agile methodologies are used, compared to traditional methodologies. The differences are shown in table 17.

Systems	Traditional system	Agile system
Project characteristics: **System architecture**	Design for current and future requirements.	Design for only what is presently essential.

Table 17 : System architecture distinction: Robustness of system architecture.

System architecture design was used as the differentiating characteristic. The results are shown in Figure 22.

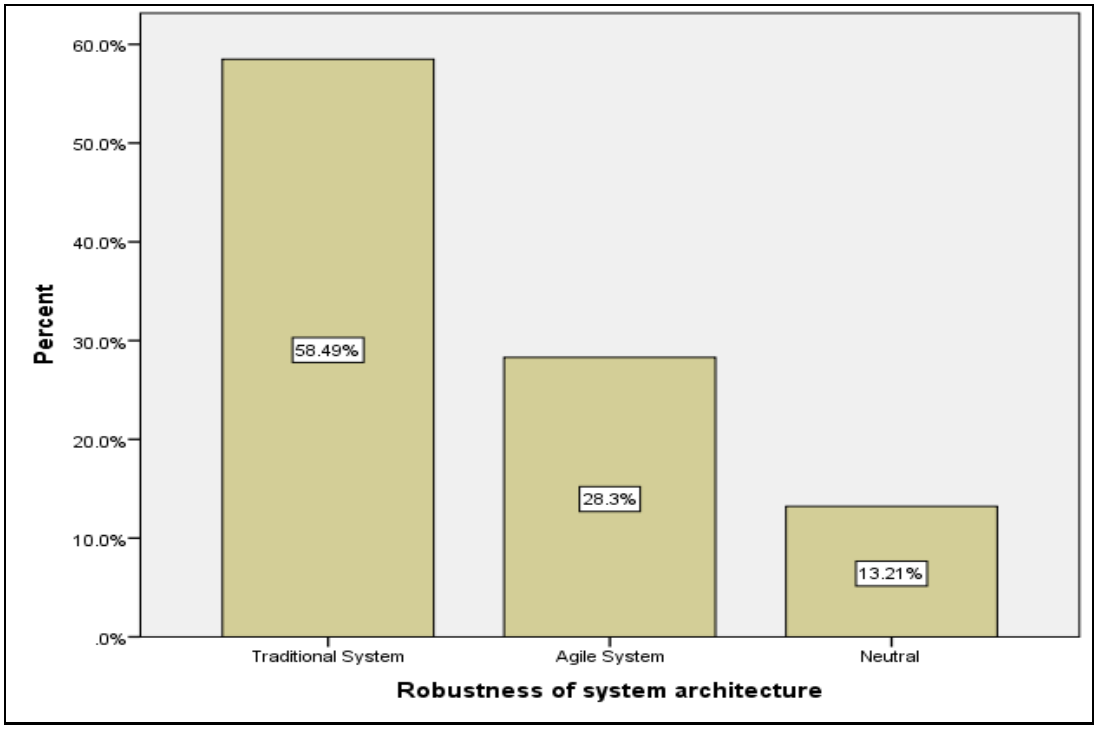

Figure 22 : Graphical representation of system architecture.

The results show that more than half of respondents (58.49%) agreed that a system's architecture will be relatively robust when a traditional methodology is used. Robustness of system architecture refers to the ability to withstand extensions with the addition of components without failing. Given the agile view of design for only what is presently essential, it is logical to concur with respondents that agile methodology use does not lead to a system with a scalable or robust architecture. The results as shown by the graph are consistent with previous findings (Cao and Ramesh, 2008; Schach, 2010).

Motivated by divergent perceptions regarding the way in which agile methodologies correlate with architecture, Breivold, Sundmark and Larsson (2010) conducted a systematic literature review seeking to understand the relation between agile methodologies and architecture.

Their conclusion was that there is lack of scientific support from an architectural benefit perspective for many of the claims made in the Agile Manifesto and that more empirical studies are needed to validate the claims.

However, to test the link between methodology use and robustness of system architecture, Fisher's exact test was conducted and a *p* value of 0.3 was returned. The *p* value in this case is greater than 0.05. The null hypothesis is therefore confirmed. The conclusion is that there is no relationship between traditional methodology use and robustness of system architecture, and secondly, that robustness of system architecture is not linked to any system development methodology.

4.3.6 Portability

This question was meant to establish the effect on *software portability* when agile methodologies are used compared to traditional methodologies. The differences are fleshed out in table 18.

Systems	Traditional system	Agile system
Project characteristics: **System architecture**	Design for current and future requirements.	Design for only what is presently essential.

Table 18 : System architecture distinction: Portability.

System architecture was used as the differentiating characteristic. The results are shown in Figure 23.

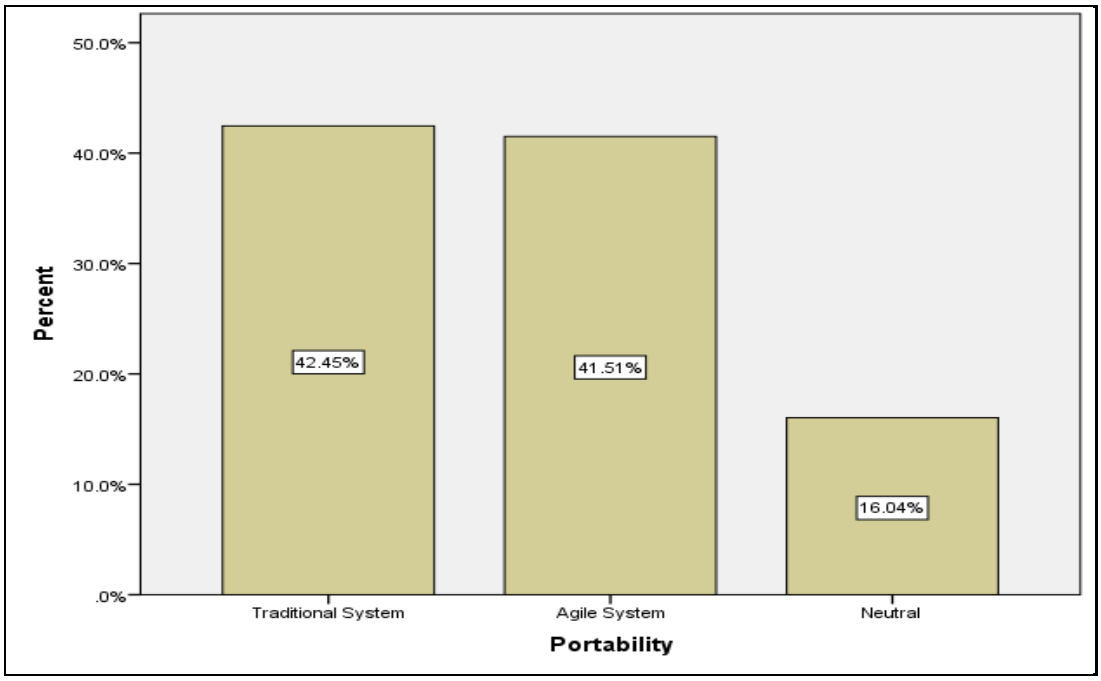

Figure 23 : Graphical representation of system portability.

The results appear to be similar, but slightly more respondents (42.4%) agreed that system portability will be relatively easier when the design is focused on current and future use. In other words, system portability will be easier if a traditional methodology is used, compared to when an agile methodology is employed. Portability refers to ability of the software to be easily installed on different hardware and software platforms.

However, to test the link between methodology use and system portability, Fisher's exact test was conducted and a *p* value of 0.15 was obtained. The *p* value in this case is greater than 0.05. The null hypothesis is therefore supported. The conclusion is that there is no relationship between methodology use and system portability and secondly that system portability is not linked to any system development methodology.

4.3.7 Meet usability needs

This question was meant to establish the effect on *system usability* when agile methodologies are used compared to traditional methodologies. The differences appear in table 19.

Systems	Traditional system	Agile system
Project characteristics: **Customer involvement.**	Customer involvement is low and passive.	• Customer is onsite and considered as a team member. • Customer involvement is active and proactive.

Table 19 : Customer involvement distinction: Meet usability needs.

The differentiating characteristic is the extent of customer involvement. The results are shown in Figure 24.

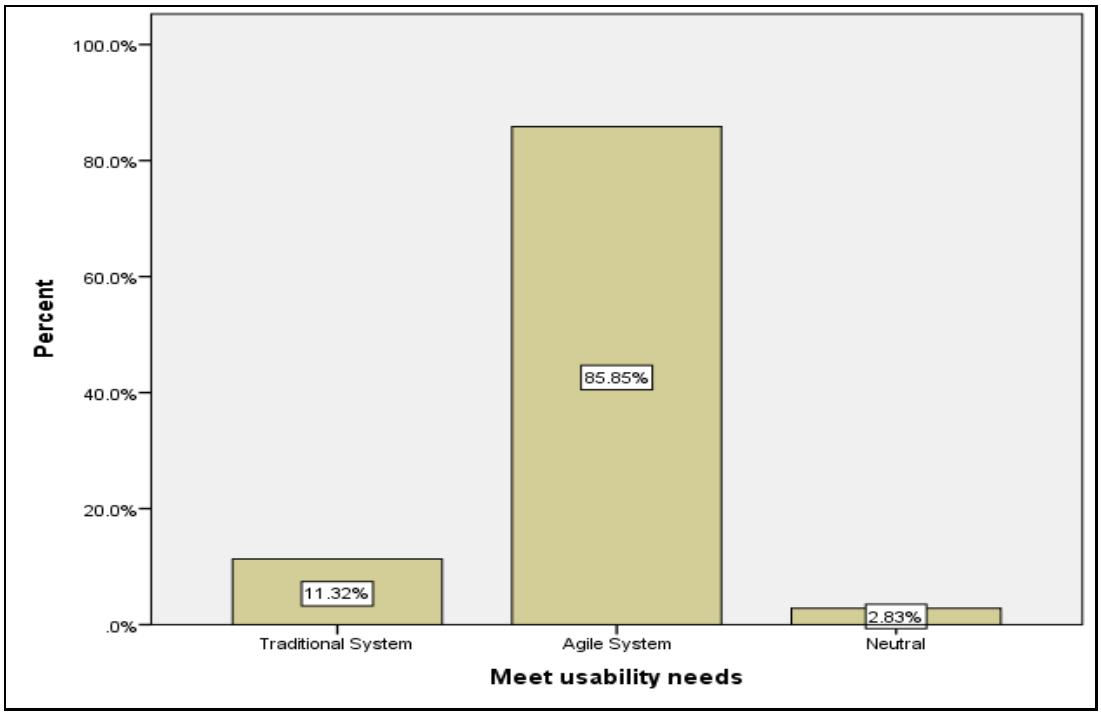

Figure 24 : Graphical representation of meet usability needs.

The results show that an overwhelming majority of respondents (85.85%) agreed that a system will meet usability needs to a greater extent when agile methodologies are used, compared to traditional system development methodologies. Agile methodologies advocate for onsite customer involvement. They also advocate for proactive customer involvement rather than reactive.

Given these views, it is logical and one would stand to reason that their use will lead to improved usability design. Previous findings have also linked agile methodology use to improved usability (McInerney and Maurer, 2005; Sy, 2007). In contrast, other authors disagree and have asserted that agile methodologies have failed or are unable to address usability issues (Jokela and Abrahamsson, 2004; Lee, 2006).

However, to test the link between methodology use and improved system usability, Fisher's exact test was conducted and a *p* value of 0.8 was obtained. The *p* value in this case is greater than 0.05. The null hypothesis is therefore corroborated. The conclusion is that, firstly, there is no relationship between agile methodology use and meet usability needs; and secondly, that meet usability needs is not linked to any system development methodology.

4.3.8 Ease of system customisation

This question was meant to establish the effect on *system customisation* in a comparative manner between agile and traditional methodology usage. The differentiating factors are shown in table 20.

Systems	Traditional system	Agile system
Project characteristics: **Customer involvement.**	Customer involvement is low and passive.	• Customer is onsite and considered as a team member. • Customer involvement is active and proactive.

Table 20 : Customer involvement distinction: Ease of customisation.

Customer involvement is the distinguishing project characteristic. The results are shown in Figure 25.

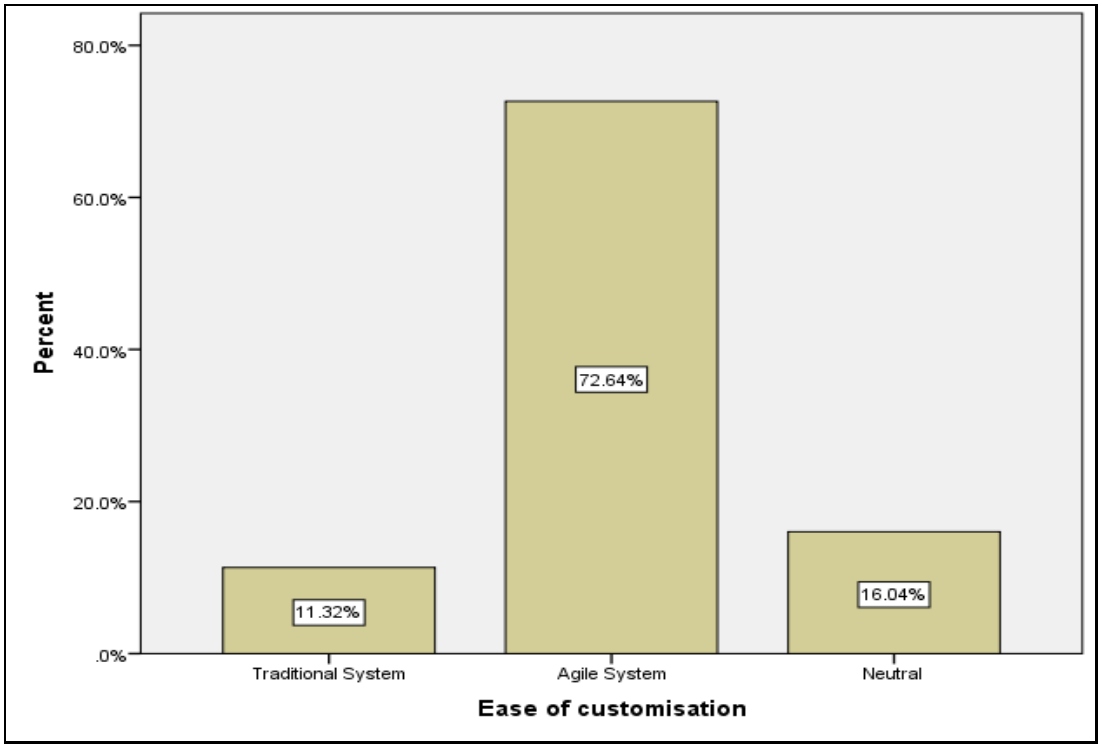

Figure 25 : Graphical representation of ease of customisation.

The results show that the majority of respondents (72.64%) agreed that a system will be relatively easy to customise when agile methodologies are used, compared to traditional system methodologies. Customisation refers to adaptation of the system to suit one's personal approach and preferences. If the customer is considered a team member, onsite and actively involved in the project as advocated by agilist, then it is reasonable to conclude that the resulting system will

be relatively easier to customise, compared to when customer involvement is low and passive (traditionalist).

However, to test the link between methodology use and ease of customisation, Fisher's exact test was conducted and a *p* value of 0.7 was obtained. The *p* value in this case is greater than 0.05. The null hypothesis is therefore upheld. The conclusion is that there is no relationship between methodology use and ease of customisation, and secondly that ease of customisation is not linked to any system development methodology.

4.3.9 Ease of system navigation

This question sought to establish the effect on *system navigation* when agile methodologies are used compared to traditional methodologies. The differences are shown in table 21.

Systems	Traditional system	Agile system
Project characteristics: **Customer involvement.**	Customer involvement is low and passive.	• Customer is on-site and considered as a team member. • Customer involvement is active and proactive.

Table 21 : Customer involvement distinction: Ease of system navigation.

The differentiating factor is extent of customer involvement. The results are shown in Figure 26.

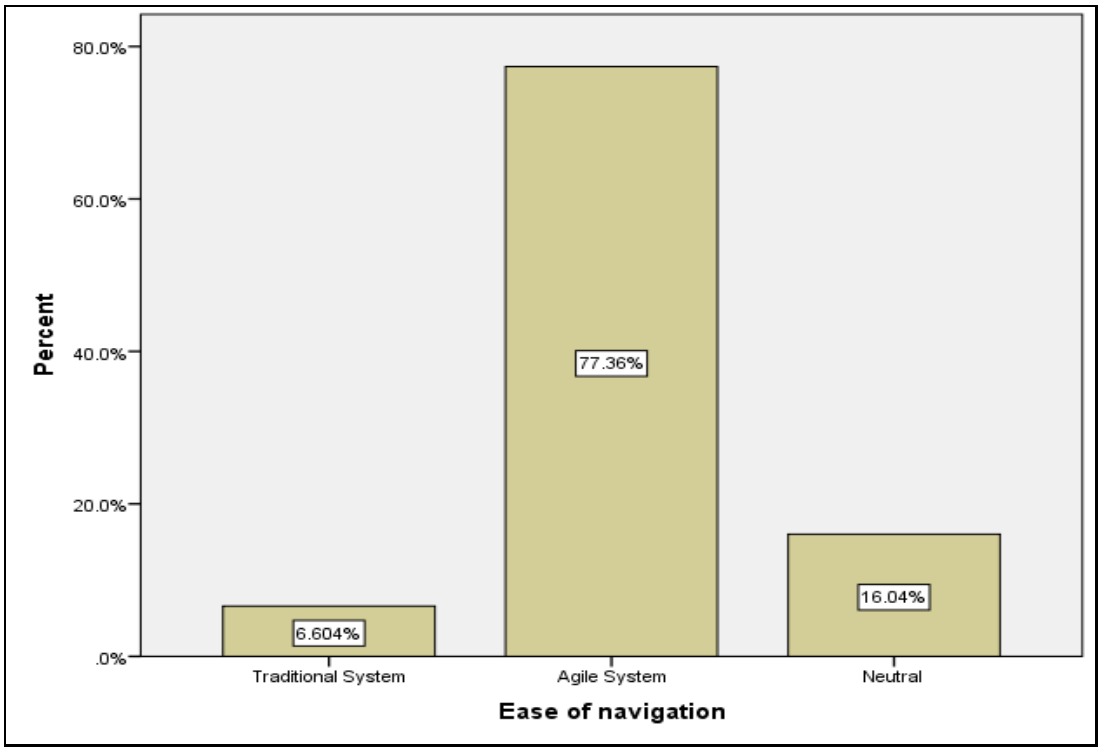

Figure 26 : *Graphical representation of ease of navigation.*

The results show that majority of respondents (77.36%) agreed that a system will be relatively easy to navigate when agile methodologies are use, when compared to traditional system methodologies. Navigation refers to user movements around the system from one page or functionality to another; again, if the customer is considered a team member, on-site and actively involved in the project, then it is reasonable to conclude that the resulting system will be somewhat easier to navigate when compared to a situation in which customer involvement is low and passive.

However, to test the link between methodology use and ease of customisation, Fisher's exact test was conducted and a *p* value of 0.6 was obtained. The *p* value in this case is greater than 0.05. The null hypothesis is therefore upheld. The conclusion is that there is no relationship between methodology use and ease of navigation, and secondly, that ease of navigation is not linked to any system development methodology.

It should be recalled from section 4.2.11 that a weak but positive relation was found between mandatory workshop attendance or training and system navigation (p=0.031). The weak nature of this relationship is something that requires further investigation.

4.3.10 Ease of system interactivity

This question was meant to establish the effect on *system interactivity* when agile methodologies are used compared to traditional methodologies. Table 22 shows the differentiating factor.

Systems	Traditional system	Agile system
Project characteristics: **Customer involvement.**	Customer involvement is low and passive.	• Customer is on-site and considered as a team member. • Customer involvement is active and proactive.

Table 22 : Customer involvement distinction: Ease of system interactivity.

Extent of customer involvement is the differentiating project characteristic. The results are shown in Figure 27.

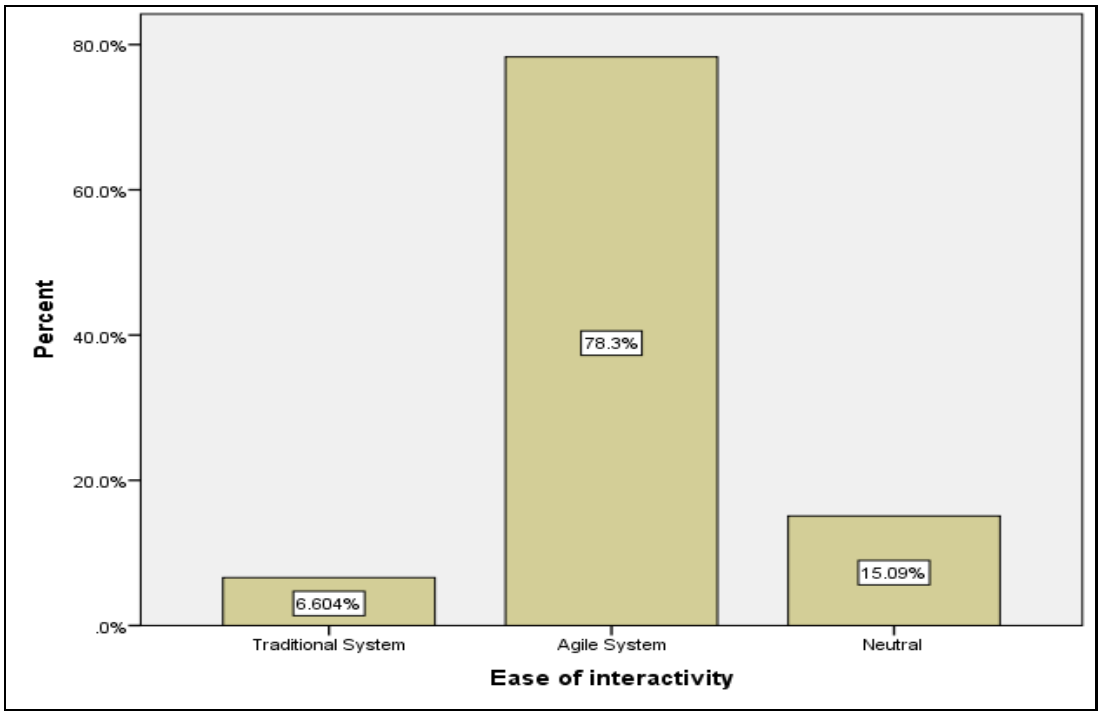

Figure 27 : Graphical representation of ease of interactivity.

The results show that majority of respondents (78.3%) agreed that a system will be relatively easy to interact with when agile methodologies are used; compared to traditional system methodologies. Interactivity refers to the extent of reciprocal action between the user and the system. If the customer is considered a team member, present onsite and actively involved in the project, then it is reasonable to conclude that the resulting system will be somewhat easier to interact with compared to when customer involvement is low, and/or passive.

However, to test the link between methodology use and ease of interactivity, Fisher's exact test was conducted and a *p* value of 1 was obtained. The *p* value, in this case, is greater than 0.05. The null hypothesis is therefore upheld. The conclusion is that there is no relationship between methodology use and ease of interactivity, and secondly, that ease of interactivity is not linked to any system development methodology.

It should be recalled from section 4.2.10 that a weak, but positive relation, was found between active stakeholder participation and system interactivity (p=0.047). The weak nature of this relationship is something that requires further investigation.

4.3.11 Error message comprehensibility

This question sought to establish the ease with which users can understand *system error messages* when agile methodologies are used compared to traditional methodologies. The differences appear in table 23.

Systems	Traditional system	Agile system
Project characteristics: **Customer involvement.**	Customer involvement is low and passive.	• Customer is on-site and considered as a team member. • Customer involvement is active and proactive.

Table 23 : Customer involvement distinction: Error message comprehensibility.

The differentiating factor is extent of customer involvement. The results are shown in Figure 28.

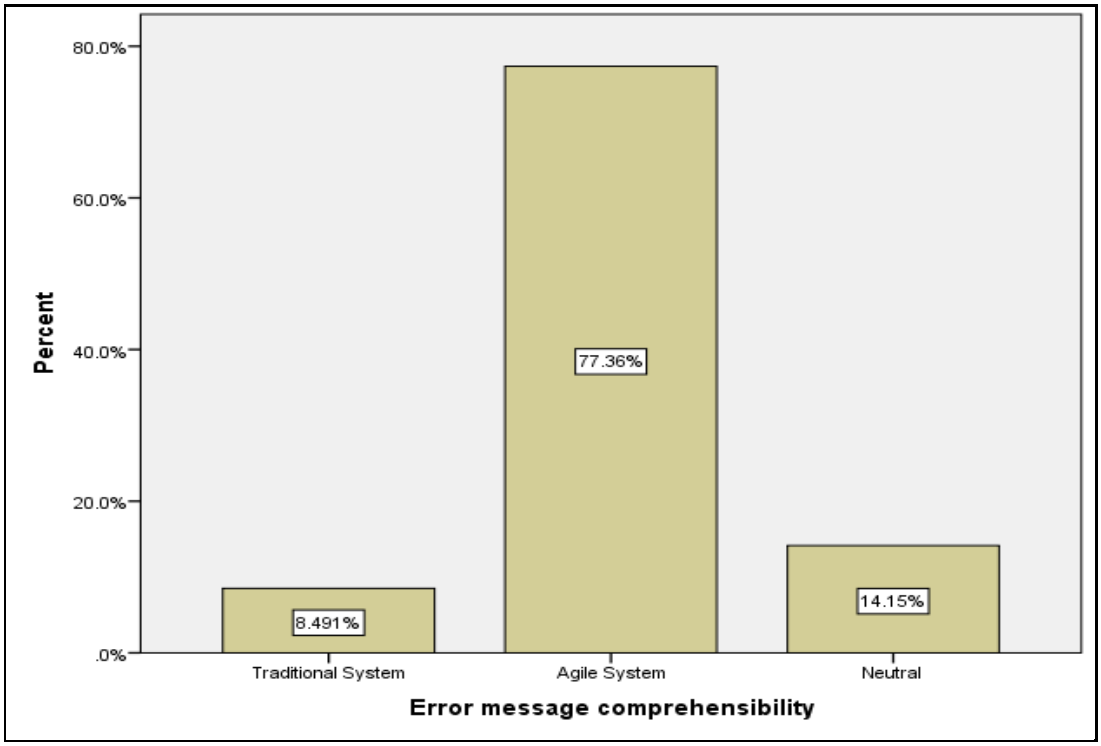

Figure 28 : Graphical representation of error message comprehensibility.

The results show that majority of respondents (77.36%) agreed that it will be relatively easier to comprehend error messages from a system developed using agile methodologies, when compared to traditional methodologies. If the customer is considered a team member, present on-site and actively involved in the project, then it is rational to concur with results shown in the graph.

However, to test the link between methodology use and error message comprehensibility, Fisher's exact test was conducted and a *p* value of 0.06 was obtained. The *p* value in this case is slightly greater than 0.05. The null hypothesis is therefore upheld. The conclusion is that there is no relationship between methodology use and error message comprehensibility, and secondly, that error message comprehensibility is not linked to any system development methodology. Because the *p* value found in this case (0.06) was slightly greater than 0.05, further investigation in this regard is recommended.

4.3.12 System help facilities

This question was meant to establish which system can produce better *help facilities* when agile system development methodologies are used, when compared to traditional methodologies. The differences appear in table 24.

Systems	Traditional system	Agile system
Project characteristics: **Customer involvement.**	Customer involvement is low and passive.	• Customer is onsite and considered as a team member. • Customer involvement is active and proactive.

Table 24 : Customer involvement distinction: Help facilities.

The differentiating characteristic is extent of customer involvement. The results are shown in Figure 29.

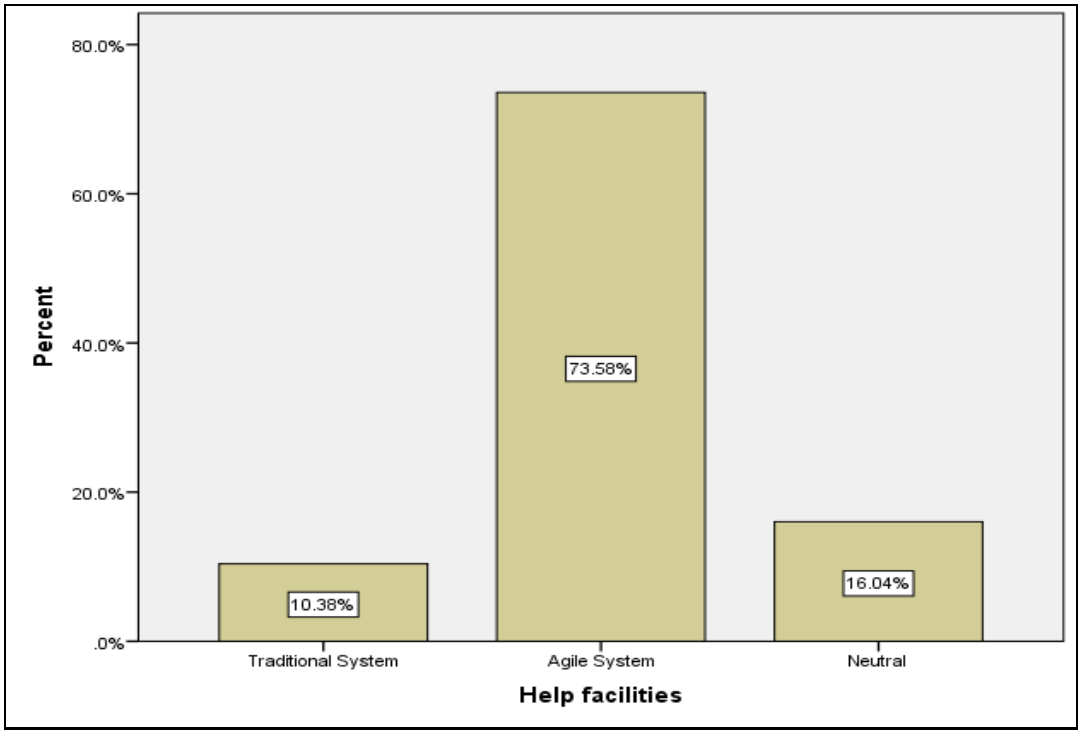

Figure 29 : Graphical representation of help facilities.

The results show that majority of respondents (73.36%) agreed that better help facilities will be produced from a system developed using agile methodologies, compared to when traditional methodologies are used. Help facilities refers to a subset of the software component that acts as documentation for the software package. Its main purpose is to explain programme function, toolbar option, or other key elements with the user interface. The help system may be located locally on the computer, or be based online. If the customer is considered a team member, present onsite and actively involved in the project, then it is rational to concur with results shown in the graph.

However, to test the link between methodology use and help facilities, Fisher's exact test was conducted and a *p* value of 0.4 was obtained. The *p* value, in this case, is greater than 0.05. The null hypothesis is therefore upheld. The conclusion is that there is no relationship between methodology use and software help facilities, and secondly, that software facilities are not linked to any system development methodology.

4.3.13 System correctness

This question was meant to establish the effect on *system correctness* when agile methodologies are used compared to traditional methodologies. The differences have been fleshed out in table 25.

Systems	Traditional system	Agile system
Project characteristics: **Requirements.**	Knowable early, largely stable, clearly defined and documented.	Emergent, rapid change, unknown and discovered during the project.

Table 25 : Requirements distinction: System correctness.

The differentiating criterion is requirements. The results are shown in Figure 30.

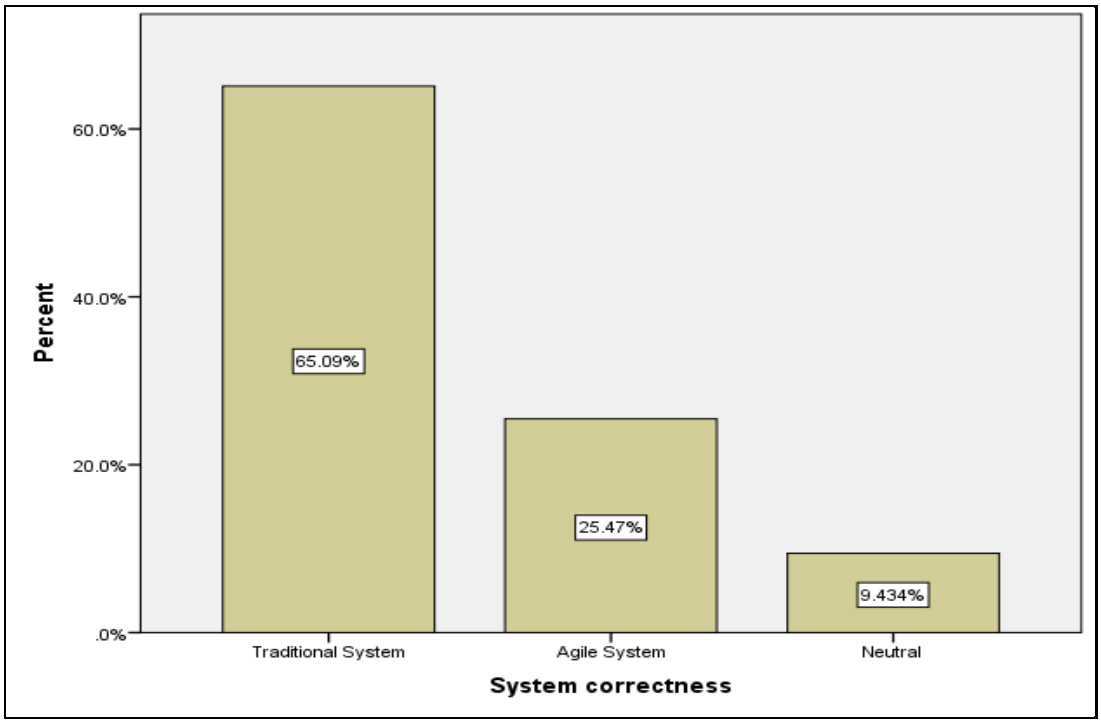

Figure 30 : Graphical representation of system correctness.

The results show that most respondents (65%) agreed that a system is likely to be more correct when traditional methodologies are used, compared to when agile methodologies are employed. System correctness refers to the ability of the system to function according to the specification or to client's needs. If requirements are knowable early, largely stable, clearly defined and documented as advocated for by traditionalists, then it is logical to concur with the results shown on the graph. Agile methodologies emphasise rapid and timely delivery of a working system, but the literature is sparse when it comes to agile methodology use and system correctness.

However, to test the link between methodology use and system correctness, Fisher's exact test was conducted and a *p* value of 0.2 was obtained. The *p* value in this case is greater than 0.05. The null hypothesis is therefore upheld. The conclusion is that there is no relationship between methodology use and system correctness, and secondly, that system correctness is not linked to any system development methodology.

4.3.14 Timeliness

This question sought to establish the effect on *timeliness* when agile methodologies are used compared to traditional methodologies. The differences are shown in table 26.

Systems	Traditional system	Agile system
Project characteristics: **Project life cycle.**	Linear/sequential model.	Evolutionary, incremental and iterative model.

Table 26 : Project life cycle distinction: Timeliness.

The differentiating factor is project life cycle model. The results are shown in Figure 31.

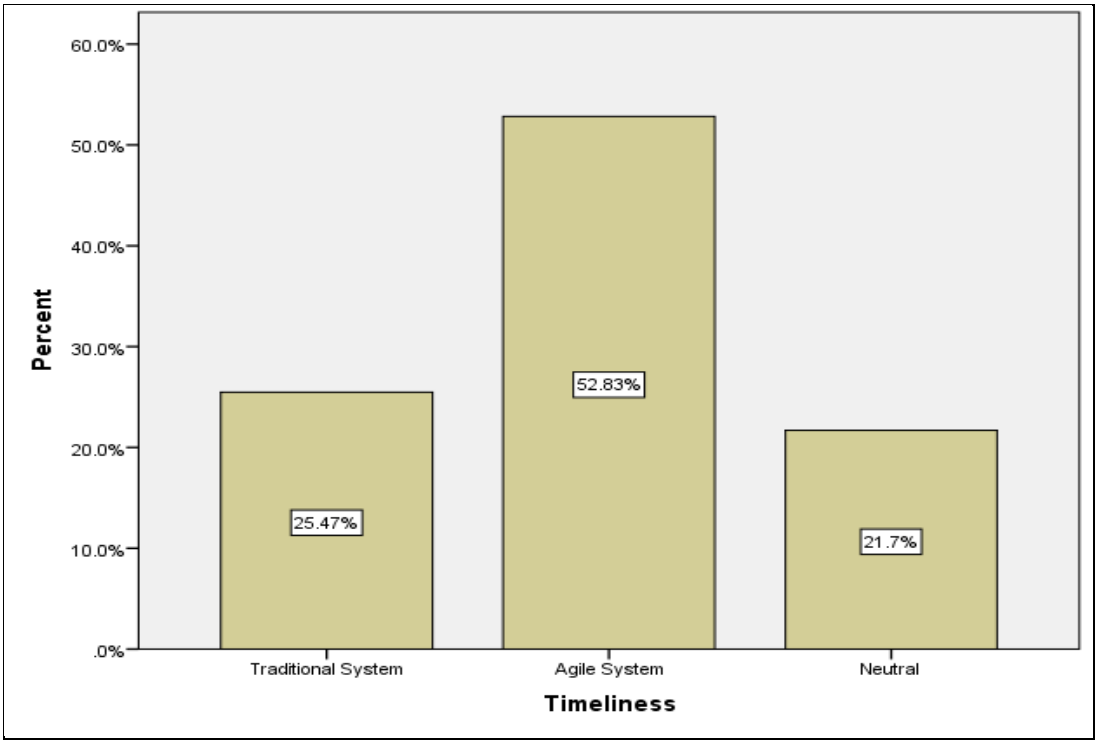

Figure 31 : Graphical representation of timeliness.

The results show that slightly more than half (52.8%) of respondents agreed that a system is likely to be delivered on time when the project life cycle is evolutionary, incremental and iterative (agile), compared to when it is linear/sequential (traditional).

Anecdotal evidence has in the past linked agile methodology use to timely delivery of software (Ahmed et al., 2010; Rao et al., 2011; Moniruzzaman and Hossain, 2013). This therefore suggests that there is not enough scientific evidence associating agile methodologies to timely delivery of software, and therefore warrants further investigation.

However, to test the link between methodology use and timeliness, Fisher's exact test was conducted and a *p* value of 0.02 was obtained. The *p* value, in this case, is less than 0.05. The null hypothesis is therefore rejected. This means that there is a significant relationship between methodology use and timeliness.

This association is linked to traditional methodologies, because the traditional methodologies cell in the SPSS cross tabulation output had a positive standard residual value of 2.9. This, in other

words, means that there were more traditional methodology respondents in that cell than the hypothesis of independence predicts (Agresti, 2007). Standardised residual values are used to show which cell or cells are contributing the most to the significance value (Agresti, 2007). Standardised residuals with a positive value mean that the cell was over-represented in the actual sample, compared to the expected frequency. Standardised residuals with a negative value indicate that the cell was under-represented in the actual sample, compared to the expected frequency (Agresti, 2007).

The corresponding Cramer's V value was 0.3, indicating weak relationship strength. The conclusion is that there is a significant relationship between traditional methodology use and timeliness. This finding has therefore contradicted previous claims (Ahmed et al., 2010; Rao et al., 2011; Moniruzzaman and Hossain, 2013) linking agile methodology use to timely delivery of software.

4.3.15 Completeness of system features

This question sought to establish the effect on *completeness of system features* when agile methodologies are used, as compared to traditional methodologies. The differences appear in table 27.

Systems	Traditional system	Agile system
Project characteristics: **Project life cycle.**	Linear/sequential model.	Evolutionary, incremental and iterative model.

Table 27 : Project life cycle distinction: Completeness of system features.

Project life cycle model is the differentiating factor. The results are shown in Figure 32.

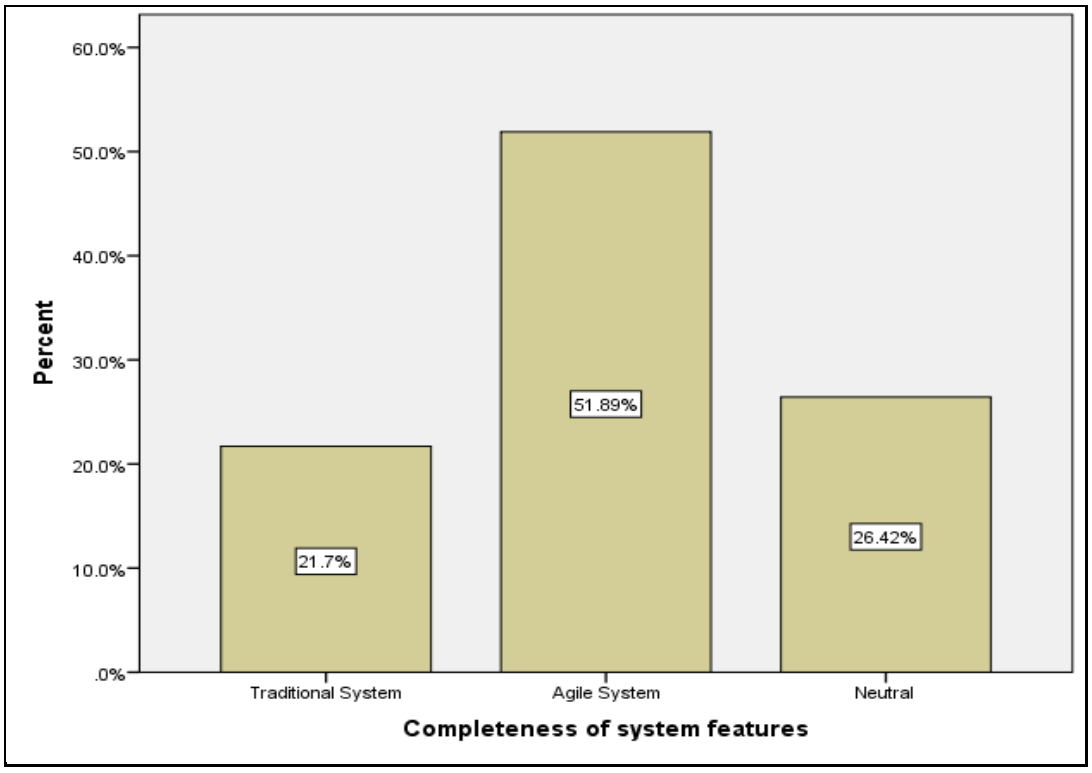

Figure 32 : Graphical representation of completeness of system features.

The results show that slightly more respondents (51.8%) agreed that a system is likely to be delivered with complete features when agile methodologies are used compared to when traditional methodologies are employed. But again, given the agile view of evolutionary, incremental and iterative development, where a system is delivered as it increasingly becomes complete i.e. evolving over time, one would logically expect that the use of agile methodologies will not lead to system that is likely to be delivered with complete features.

Paradoxically, the graph shows that the use of agile methodologies will lead to a system that will be delivered with complete features, when compared to traditional methodologies.

However, to test the link between methodology use and completeness of system features, Fisher's exact test was conducted, and a *p* value of 0.7 was obtained. The *p* value in this case is greater than 0.05. The null hypothesis is therefore upheld.

The conclusion is that there is no relationship between methodology use and completeness of system features, and secondly, that completeness of system features is not linked to any system development methodology.

4.3.16 The most suitable project life cycle model

Finally, this question was meant to establish which *project life cycle model* was most suitable for projects on which respondents worked. The results are shown in Figure 33.

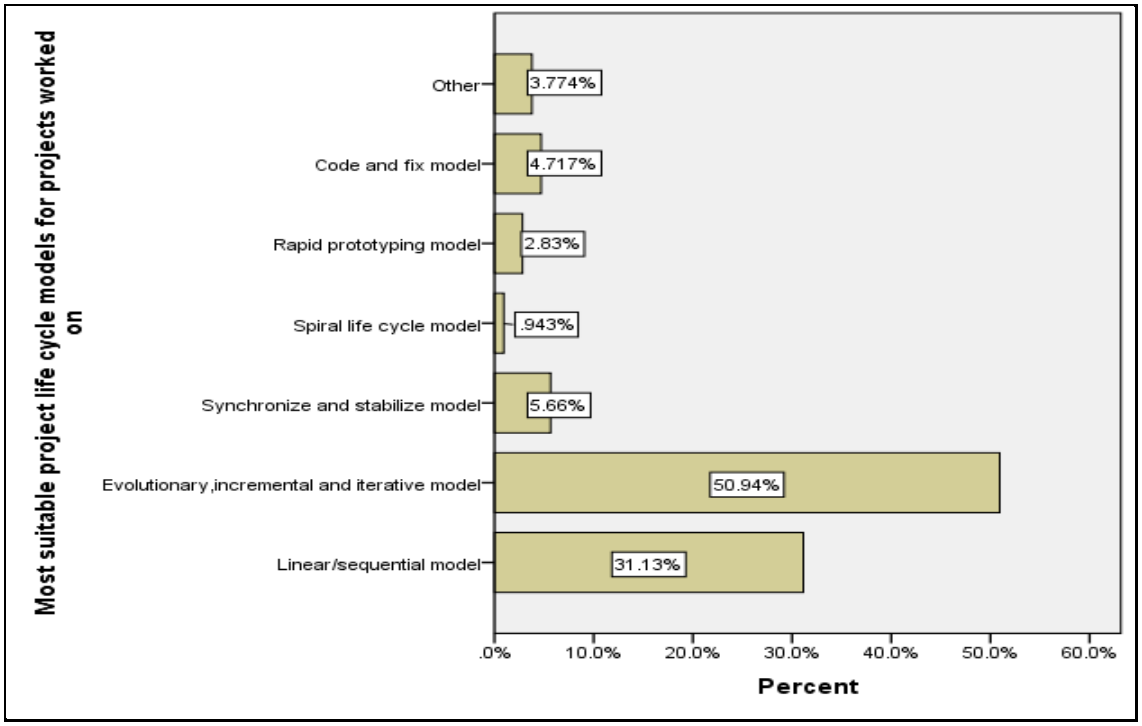

Figure 33 : Graphical representation for most suitable project life cycle model.

The results show that about half of respondents (50.94%) selected evolutionary, incremental and iterative model as the most suitable project life cycle model for the projects on which they worked. The second most suitable model was linear sequential.

Even though an evolutionary, incremental and iterative approach is emphasised by agile methodologies, it would be misleading to conclude that the most suitable project life cycle model is agile. It has been argued before (see section 4.2.2) that iteration and incrementation are not features unique to agile methodologies (Jiang and Eberlein 2009; Rao et al., 2011). Iteration means the first version of a software artefact is produced, and then revised to produce the second, and so on. Incrementation means producing a subset of what the software will achieve, and gradually adding other core functionality until it becomes complete (Schach, 2010).

Despite the fact that steps in the waterfall model seem sequential, one would turn or be tempted to interpret or assume that software development with this model occurs in discrete phases, but in practice, it is considerably different (Schach, 2010). The same author advances two reasons for this; first software professionals are human, and therefore subject to making errors. Second,

the client's requirement change in the course of the development process. Therefore, in an ideal world, software is not developed as sequentially or linearly depicted by the waterfall model (Schach, 2010). Iteration and incrementation are an intrinsic aspect of software development, and have been used in software engineering for over four decades (Larman and Basili, 2003).

4.4 Discussion and Conclusion

This chapter was an in-depth and critical analysis of survey results. The analysis philosophy, choice of methods and type of test employed was justified. Evidence of scale reliability was also articulated. The results show that combinations of traditional and agile methodologies are more frequently used than any single methodology. This suggests that most organisations are not locked into one methodology type, but employ the best features of each. Even though iterative and incremental development was selected by most respondents as the meaning of agility, it would be inappropriate and somewhat inaccurate to conclude that it is tantamount to agility, because it is not a feature unique to agile methodologies alone. To define agile methodologies as iterative and incremental development alone would be an incomplete analysis. The majority (85%) of respondents did not agree with all core principles of the Agile Manifesto. This means that a suitable common trend to enable the derivation of an appropriate consensus definition of agility did not emerge or observed. Furthermore, the results uncovered that Scrum was the most widely (60%) used, where agile methodology thus by far corroborating previous findings (Conboy 2009; Rao et al., 2011; Moniruzzaman and Hossain, 2013). Contrary to previous studies, the much talked about extreme programming (XP) of being the main pillar and representative agile methodology obtained the lowest score (1.2%). This might be the beginning of a new trend that needs to be monitored and confirmed. The overall adoption of XP might have been hampered by its overly prescriptive nature on the use of techniques as obligations, rather than as options (Meyer, 2014). Recent trends suggest that Scrum has now taken over as the representative agile methodology (Rao et al., 2011; Moniruzzaman and Hossain, 2013; Kapitsaki and Christou, 2014).

It was also observed that agile methodologies are increasingly gaining popularity with most organisations transitioning to them. Despite reports of widespread adoption (Salo and Abrahamsson 2008; Zhang et al., 2010; Laanti et al., 2011), Rodriguez et al. (2012) have cautioned that most publications are not academic, but rather written by agile consultants themselves, along with tool vendors, professional societies, market research organisations, indicating a potential conflict of interest that may threaten trustworthiness and validity of results. Still on agile adoption, the results show that most organisations are still at a level of between 1-

5 years with regards to agile experience, where only a few organisations have past the experience level of six years and beyond.

The inquiry on agile team size, agile project size and agile organisational size yielded perfectly consistent results, further indicating that the scale used was internally consistent. Further to that, the results also revealed that agile methodologies work well in small environments and are unsuitable for large environments. This conclusion is perfectly in line with previous studies (Dyba and Dingsoyr, 2008a; Abrahamsson et al., 2010; Barlow et al., 2011; Saxena and Kaushik, 2013). In addition, it was uncovered that 'other' meaning customised SDLC, Agile or in-house methodologies were more prominent when the environment gets increasingly large or complex, suggesting that customised methodologies are more suitable for complex or larger environments.

With regards to quality standards, the most used quality standard was the ISO 9000. It was also evident that approximately 40% of organisations surveyed are not employing any software standards in their system development processes.

The results further show that approximately 40% of agile users are not employing any software quality standards, while 60% are employing software quality standards. This observation raises questions about claims that agile methodologies have built-in software quality abilities (Huo et al., 2004; Bhasin, 2012). This leaves one to wonder whether previous reports linking agile methodologies to improved software quality (Santos et al., 2011; Rao et al., 2011; Barlow et al., 2011) are as a result of using pure agile methods with built-in software quality abilities, or as a result of external quality standards employed. The question as to why agile users might still use quality standards if their methodology has built-in software quality abilities is contentious, and warrants further investigation.

Most respondents (75%) agreed that key stakeholders actively participated on the projects that they worked on, and further to that, albeit weak, a significant relationship was found between active stakeholder participation and ease of system interactivity.

Relative to those who disagreed (17%), it was found that about half (49%) of respondents attended mandatory workshops and training, while 34% remained neutral. Further to that, albeit weak, a significant relationship was found between attendance of workshops or training, and ease of system navigation.

With regards to relevant domain knowledge, it was evident from respondents' professions namely: project managers; programme or portfolio managers; business/system analysts; software architects; IT lecturers; quality assurance consultants; software testing analyst; developers/programmers; process analyst; IT consultants and business stakeholders, that they had the relevant and necessary domain knowledge to complete the survey in a manner that can produce meaningful results. In terms of respondent's country of residence, although dominated by respondents from South Africa, it was evident from the results that representation was global, encompassing all major continents of the globe.

From an analysis of research variable with regards to methodology use, correlational values are summarised in Table 28.

#	Research variable	*P* values (Fisher's exact test)	Correlation with traditional methodology use	Correlation with agile methodology use
1.	Ease of system maintenance.	0.97	No	No
2.	Ease of system testing.	**0.014**	**Yes**	No
3.	Ease of system training.	0.7	No	No
4.	Ease of system learning.	0.25	No	No
5.	Robustness of system architecture.	0.3	No	No
6.	Portability.	0.15	No	No
7.	Meet usability needs.	0.8	No	No
8.	Ease of system customisation.	0.7	No	No
9.	Ease of system navigation.	0.6	No	No
10.	Ease of system interactivity.	1	No	No
11.	System correctness.	0.2	No	No
12.	Timeliness.	**0.02**	**Yes**	No
13.	Completeness of system features.	0.7	No	No
14.	Error message comprehensibility.	**0.06(close)**	No	No
15.	System help facilities.	0.4	No	No

Table 28 : Research variable correlation matrix.

The matrix (Table 28) shows that there is a correlation between traditional methodology use and ease system testing, as well as with traditional methodology use and timeliness. The case of system testing is not sufficient to conclude, because software quality standards (extraneous variable) also had a significant correlation with ease of system testing (p=0.017). Further investigation is recommended.

With regards to timeliness, albeit weak, the conclusion is that there is a significant relationship between traditional methodology use and timeliness. This is a contradiction relative to previous reports, putting forward anecdotal evidence and linking agile methodologies to timely delivery of software (Ahmed et al., 2010; Rao et al., 2011; Moniruzzaman and Hossain, 2013).

No correlation was found between agile methodology use and any of the software quality parameters identified for this study.

Generally, as shown in Table 28, this study did not find a significant correlation between methodology use and the following software quality parameters: system maintenance; system training; system learning; system architecture; portability; meeting usability needs; system customisation; system navigation; system interactivity; error message comprehensibility; help facilities; system correctness; and completeness of system features. There was therefore insufficient evidence linking agile methodology use to improved software quality for the just mentioned software quality parameters.

Therefore, the first major conclusion is that this research failed to link methodology use to most of the selected software quality criteria. The fundamental deduction here is that even though possible, it is difficult to link software quality criteria to a methodology. The reason for the above is threefold: firstly, software itself is inherently complex and the development process itself is complex (Brooks, 1987), with significant amount of variables and dynamics. This therefore makes it highly improbable to conclusively link a particular quality criterion to the use of a particular methodology (Avison and Fitzgerald, 2006). The reason for this is that it is not possible in a software development environment to hold all other variables constant, as with clinical laboratory experiments. Secondly, quality itself is a highly subjective, complex and multifaceted concept that is difficult to define, where an agreed upon and all-encompassing definition of software quality is still to arrive. Thirdly, methodologies as promulgated by their authors are never quite used as intended. All the quality enhancing issues promised by the methodologies seem convincing, but regrettably only exist in theory. Practically speaking, they are rarely applied (Avison and Fitzgerald, 2006). This same sentiment has been shared by other consultants, asserting that 50% of their client base claims to be using agile methodologies, but only 10% of the work is actually being done in an agile manner (Bedell, 2011). In addition to that, all the quality principles theoretically promised and prophesied by given methodologies assume that the software development process unfolds mechanistically, but pragmatically, it is considerably different, and driven by humans. This fundamental disconnect between theory and application makes it difficult to prove that a particular methodology led to A, B, C etc.

The second major conclusion is that agile methodologies, when compared to traditional methodologies, do not improve software quality for the selected criteria espoused by this study. Several studies have reported broad benefits of improved software quality (Santos et al., 2011; Rao et al., 2011; Barlow et al., 2011) but other studies have raised scepticism and argued that empirical evidence is conjectural and lacking (Ambler, 2008; Selic, 2009; Rodríguez, et al., 2009;

Abrahamsson et al., 2010; Barlow et al., 2011; Hummel, 2014). Claims that agile methodologies use leads to improved software quality could not be validated for selected quality criteria.

The third and last major conclusion is that, in general this research has so far refuted findings from previous research (citations were provided), it has also upheld and corroborated findings of previous research, and, more importantly, it has also made new significant findings on its own. As referred to in the previous chapter (research design and methodology), the mode of reasoning underlying this research is deduction. Deduction, which is associated with the Scientific method, is a form of logic that draws conclusions from other premises or statements that follow from such premises. Simply stated, it draws conclusions from a particular case, based on the general case i.e. using a top-down approach (Mouton, 2005).

Therefore, against the aforesaid background, results of this study can be generalised. Generalisation, also known as external validity, in this case means that these research findings can be extended to other context, or areas beyond the original context where the study was carried out. The researcher is confident that if this study was to be repeated in a given form, the results would not deviate significantly from those contained herein.

CHAPTER 5: CONCLUSION

5.1 Introduction

The main goal of this chapter is to synthesise the dissertation and validate that the research question was answered and research objectives achieved. The chapter is made up of eight major sections. The first section is an introduction of the purpose and contents of the chapter. The second section is a synopsis of dissertation chapters. The third section is a confirmation that the research question was answered and research objectives met, while the fourth articulates contributions of the research to practice and academia. Like any other research effort, the fifth section is an acknowledgement of research limitations. Recommendations and areas for future research are discussed in the sixth section, where a final section provides the conclusion to the study. The last section is a personal reflection on the subject of the research by the researcher.

5.2 Synopsis of chapters

Chapter one provided extensive contextualisation and articulation of the research problem and background. It also formalised the theoretical and definitional framework. The following five areas ambiguous in existing literature, led to problem statement formulation and study motivation:

(a) An over-emphasis of existing research on the system development process rather than the product (March and Smith 1995; Abrahamson et al., 2009; Bhasin, 2012); (b) limited studies on agile methodologies in the quality assurance arena (Bhasin, 2012); (c) huge and growing controversy as well as scepticisms regarding the perceived benefits of agile methodologies compared to traditional methodologies (Dyba and Dingsoyr, 2008b; Ambler, 2008; Rodríguez et al., 2009; Selic, 2009; Abrahamsson et al., 2010; Barlow et al., 2011; Hummel, 2014); (d) Sparsity of global studies investigating the perceived benefits of agile methodologies compared to traditional methodologies at the time of literature review; (e) the need for more empirical studies on agile methodologies and software engineering in general (Perry et al., 2000; Dyba and Dingsoyr, 2008a; Zhang et al., 2010)

The main research question sought to understand the impact of the use of agile methodologies on selected software quality criteria compared to traditional methodologies. The main research question was then disintegrated into sub-questions that could be easily answered by the research methodology employed. Following from the prior, the research objectives were therefore to ascertain the way in which selected quality parameters are impacted regarding methodology use in a comparative manner. Similarly, the main research objective was then sub-divided into objectives that the study sought to achieve.

Key research variables i.e. agile methodologies, traditional methodologies and software quality were identified and their definitions articulated and operationalised. Taking cognisance of the different and diverse criteria for software quality, a dichotomy was articulated from the word 'software quality', viz. that between 'system quality' and 'information quality', respectively (Delone and McLean, 1992). The Delone and McLean (2003) model of IS success was therefore used as a theoretical framework to derive the quality criteria used for this research.

Chapter two deployed a critical commentary and extensive literature review, relevant to the study, with the objective of identifying a gap, weaknesses or flaws in the already existing body of knowledge on agile methodologies from a software quality enhancing perspective. The Chapter commenced by presenting an overview of traditional methodologies, the most popular ones identified were the Waterfall Model, the Spiral Model, the V Model and the Rationale Unified Process. Their current adoption trends were also discussed. It was evident from the literature that they remain the dominant methodologies in use (Zhang et al., 2010; Kendall et al., 2012; Moniruzzaman and Hossain, 2013).

Next was an overview of agile methodologies, the most popular ones identified were Scrum, Extreme Programming (XP), Lean Development, Feature Driven development, Dynamic software development methods, Crystal Methodologies and Adaptive Software Development. Their Current adoption trends were also discussed. It was evident from literature that they are becoming popular with the number of organisations embracing them increasing. (Schwaber et al., 2007; Salo and Abrahamsson, 2008; Zhang et al., 2010; Esfahani et al., 2010; Laanti et al., 2011).

Despite the growing number of organisations transitioning to agile, it was highlighted that several studies have found that there are many problems, impediments and difficulties encountered with agile adoption, as well as putting the proposed benefits into practice (Barlow et al., 2011; Leau et al., 2012; Prause and Durdik, 2012; Asnawi et al., 2012; Pathak and Saha, 2013; Twidale and Nichols, 2013).

A discussion on the historical underpinnings of agile methodologies was also explored. Some literatures sources have argued that there is significant evidence that practices used in both traditional and agile methods have historical links, and that many practices in both methods have roots in other disciplines, as well as in traditional engineering principles (Larman and Basili, 2003; Merisalo-Rantanen et al., 2005; Meso and Jain, 2006; Jian and Eberlein, 2009). On the contrary, sceptics argue that the Agile Manifesto principles, as well as agile methodologies, are

insufficiently grounded in theory, with no readily observable or agreed upon definition (Conboy and Fitzgerald, 2004; Dingsoyr et al., 2012; Aitken and Ilango, 2013; Hummel, 2014).

The discussion above was followed by a critical review of related work. Gaps in existing studies were eminent, and apparent in the following areas: (a) some related studies reported that agile methods do improve product quality, however, some studies were inconclusive, while others were contradictory (Dyba and Dingsoyr, 2008b); (b) some studies were merely subjective opinions of the authors, with expert analysis, evaluations and contributions. Generalisations could not be made, because no rigorous research methodology was employed (Huo et al., 2004; Mnkandla and Dwolatzky, 2006); (c) limitations of most studies were never acknowledged, despite inherent shortcomings with all research (Ahmed et al., 2010); (d) Research variables were not properly defined, and operationalised (Abbas et al., 2010). Conclusions drawn from such research lacked credibility, because respondents are left to interpret research variables in their own way; (e) broad conclusions were drawn, that the use of agile methodologies leads to improved product quality without explicitly stating the software quality metrics selected (Ahmed et al., 2010); (f) the reviewed studies were mostly carried out in small, isolated settings with no global study at the time of doing the literature review (Barlow et al., 2011); (g) with most studies, the research methodology employed was qualitative (Hummel, 2014); (h) some of the studies were success stories, and lessons were learnt with only limited contextual information (Rodríguez et al., 2009). Therefore, the opportunity to carry out this study was primarily built on the just articulated shortcomings, or defects in existing literature.

Chapter three unpacked the research design and methodology employed. The choice of methods used was explained and justified. The philosophical paradigm underlying the research was positivism. The way the problem statement was formulated so as to be answered revealed that the research is explanatory in nature. It was disclosed that the research strategy is a survey using a structured questionnaire as the data collection method. Furthermore, it was stated that the data analysis approach is quantitative, using deduction as the mode of reasoning. The sampling frame was also identified, and its composition discussed. The fundamental research design underlying this research was causal comparative, as well as correlational. Validity, reliability issues, mitigation measures and ethical considerations were also addressed in this chapter. The chapter then proceeded by highlighting the fact that quantitative and qualitative methods or approaches represent an interactive continuum, rather than bi polar opposites, adding that studies these days tend to find themselves on the quantitative/qualitative interactive continuum. Following on from the former, it was underscored that this study lies within that interactive continuum, but that it

leans more towards the quantitative paradigm (Newman and Benz, 1998; Onwuegbuzie and Leech, 2005).

Chapter four was a comprehensive analysis of data collected from the survey. The specific tests applied were described, and their choice justified, primarily owing to fact that the data type was categorical (nominal). SPSS was used to analyse the data. Statistical techniques employed were frequency and correlational analysis. Reliability of the scale was further demonstrated by computation of Cronbach's alpha coefficient. Demographic and background data was analysed using frequency analysis, aided by bar charts as graphical models. Analysis of research variables was done by applying a test of significance, using the fundamental assumption of the null hypothesis as the point of departure. Specifically, bar graphs were employed so as to show the number of occurrences in each category. Fisher's exact test was conducted to ascertain the association between selected software quality parameters and methodology use. Cramer's V value was computed so as to establish the strength of significant relationships. The following were main findings:

5.2.1 Traditional and agile methodologies combined are being used (47%) more than any other methodology;

5.2.2 Agile methodology use (28%) surpassed traditional methodology use (19%);

5.2.3 A suitable consensus definition for agile methodologies did not emerge from the data collected;

5.2.4 Scrum was the most widely used agile methodology by far;

5.2.5 The popularity and adoption state of XP showed a significantly decreasing trend;

5.2.6 In terms of agile team and organisational experience, the average results showed that 87% of organisations are between 1-5 years old, while 13% have an experience of 6 years and beyond;

5.2.7 The inquiry on agile team size, agile project size and agile organisational size yielded perfectly consistent results, and showed that agile methodologies are suitable for small projects or environments and need adaptations for large and complex environments or projects;

5.2.8 The results showed that 'Other' methodologies, i.e. customised agile or SDLC, are suitable, as the environment increasingly becomes larger and more complex;

5.2.9 It was uncovered that 40% of organisations do not use any software quality standards or frameworks;

5.2.10 It was apparent that 60% of agile methodology users are still employing software quality standards, despites claims that agile methodologies have built-in software quality standards;

5.2.11 Comparatively, ISO 9000 was the most widely used (31%) software quality standard;

5.2.12 There was 75% active stakeholder participation on projects;

5.2.13 Approximately half of project stakeholders attended mandatory workshops/training, while half did not attend;

5.2.14 Significant positive relationships were found between *ease of system testing(p=0.014)*, *timeliness(p=0.02)* and traditional methodology use;

5.2.15 A significant positive relationship was found between software quality standard used and *ease of system testing(p=0.017)*;

5.2.16 A significant positive relationship was found between active stakeholder participation and *ease of system interactivity(p=0.047)*;

5.2.17 A significant positive relationship was found between active workshop attendance /training and ease of system navigation(p=0.031);

5.2.18 The association between any of the selected software quality criteria in relation to agile methodology use was not apparent;

5.2.19 The association between most of the selected software quality criteria in relation to methodology use in general was also not apparent;

5.2.20 It is possible, but not easy, to conclusively link a software quality criterion to a methodology, because of the ramifications and complexities of the software development process, and also because of the subjective and multifaceted nature of quality;

5.2.21 The most suitable project life cycle model was evolutionary, incremental and iterative;

5.2.22 It was implicitly uncovered that methodologies are rarely used as originally intended by their authors, highlighting a fundamental misalignment or disconnect between theory and practice.

5.3 The research question answered

The main research question was: **what is the impact of the use of agile methodologies on software quality criteria compared to traditional methodologies?**

***Sub question 1*:** What is the impact of the use of agile methodologies on system quality criteria compared to traditional methodologies?

Sub question 2*:* What is the impact of the use of agile methodologies on information quality criteria compared to traditional methodologies?

The main research objective was: **to investigate if a positive correlation exists between system and information quality criteria when agile methodologies are used compared to traditional methodologies.**

Sub-objective 1*:* To ascertain if a positive correlation exist between the following system quality criteria when agile methodologies are used compared to traditional methodologies: ease of system maintenance; ease of system testing; ease of system training; ease of system learning; robustness of system architecture; portability; meet usability needs; ease of system customisation; ease of system navigation; ease of system interactivity; system correctness; timeliness and completeness of system features.

Sub-objective 2*:* To ascertain whether a positive correlation exists between the following information quality criteria when agile methodologies are used, when compared to traditional methodologies: error message comprehensibility and system help facilities.
The research question was answered and objectives met. Evidence of this assertion is summarised in Table 29. Therefore, all the research questions raised at the beginning of this study were answered.

#	Research variable	*P* values(Fisher's exact test)	Correlation with traditional methodology use	Correlation with agile methodology use	Research question answered
1.	Ease of system maintenance.	0.97	No	No	Yes
2.	Ease of system testing.	**0.014**	**Yes**	No	Yes
3.	Ease of system training.	0.7	No	No	Yes
4.	Ease of system learning.	0.25	No	No	Yes
5.	Robustness of system architecture.	0.3	No	No	Yes
6.	Portability.	0.15	No	No	Yes
7.	Meet usability needs.	0.8	No	No	Yes
8.	Ease of system customisation.	0.7	No	No	Yes
9.	Ease of system navigation.	0.6	No	No	Yes
10.	Ease of system interactivity.	1	No	No	Yes
11.	System correctness.	0.2	No	No	Yes
12.	Timeliness.	**0.02**	**Yes**	No	Yes
13.	Completeness of system features.	0.7	No	No	Yes
14.	Error message comprehensibility.	**0.06(close)**	No	No	Yes
15.	System help facilities.	0.4	No	No	Yes

Table 29 : Research question answered matrix.

5.4 Significance and contributions of the research

This research made the following contributions to practice and academia:

5.4.1 The main contribution of this research is the 22 major findings discussed in section 5.2.

5.4.2 This research validated and contradicted existing knowledge or facts. The contradiction of previous knowledge stimulates debate and provides opportunities for further research as discussed in section 5.6.

5.4.3 The research findings made a contribution to the epistemology of Information systems, Software project management, Software engineering and Computing in general.

5.5 Research limitations

Inherent in all research are limitations. The following is an acknowledgement of research shortcomings:

5.5.1 In cases where significant relationships were found, temptations to infer causation or causality will be premature. This means that conclusions cannot be drawn, that A causes B in such cases. Correlational and causal comparative research design, as is the case with this research, is weak at drawing causality conclusions (Neil, 2000).

5.5.2 The questionnaire was designed with the intention of querying respondents, to answer the questions according to their practical project experiences, observations and lessons learnt with regards to software quality and methodology use. This might not have been the case. Some respondents might have answered the questions based only on the theoretical distinctions between traditional and agile methodologies.

5.5.3 Every effort was made by the researcher to control for extraneous variables, despite this control endeavour, some elusive and clandestine variables might have come into play, suppressing significant relationships that would have been otherwise be uncovered.

Despite these limitations, the research results are still valid and valuable to both practitioners, for application and academics for further research.

5.6 Recommendations for future research

5.6.1 The need for a better comprehension of agility as well as common standards for agile.

For a concept to be properly investigated, it is imperative that it has a strong underpinning logic and rationale. The Agile Manifesto postulated its principles and philosophy. Other authors have also defined agile mostly as being an incremental and iterative paradigm (Leau et al., 2012; Moniruzzaman and Hossain, 2013; Mohammad et al., 2013, but this incremental

and iterative feature is not tantamount to agility (Larman and Basili, 2003; Jiang and Eberlein 2009; Rao et al., 2011). They have therefore been diverse interpretations of what it means, also evident from the results of this research. Most interpretations tend to be contextual. It therefore becomes a daunting task trying to investigate a concept that is not properly defined or clarified. Even though there are some common elements or features in all agile methodologies, consensus on a consistent definition has not convincingly emerged. The emergence of a common definitional framework for agility is necessary to build a meaningful cumulative and coherent body of knowledge.

5.6.2 The need to develop a unifying framework for agile methodologies.

This recommendation is inextricably linked to the first. Several agile methodologies exist, all postulating different set of practices regarding how to build software systems. They all have strengths and weaknesses. They have also been competition amongst them in terms of popularity and suitability in different environments. Building on a common definitional framework for agile methodologies, the very best features of each should be leveraged upon and combined into a consolidated, integrated and unifying meta-framework. A typical or proposed research topic here can be formulated as illustrated below.

'Agile methodologies: Towards a unifying meta-framework or methodology'

5.6.3 The need to develop a unifying framework for traditional methodologies.

Prior to the inception of agile, traditional methodologies were the dominant paradigms for developing software systems. Many of them exist and their weaknesses have also been uncovered and well-documented. Their definition is not so ambiguous compared to agile methodologies. The very best features of each should be leveraged upon and combined into a consolidated, integrated and unifying meta-framework. A typical or proposed research topic here can be formulated as illustrated below.

'Traditional methodologies: Towards a unifying meta-framework or methodology'

5.6.4 The need for the development of a unified and sufficient software quality model.

Several models exist to help improve the software development process, irrespective of methodology used, such as the capability maturity model (CMM), project management maturity model (PMMM), ISO 9000 amongst others. They all have weaknesses and strengths. The very best features of each should be leveraged upon and combined into a consolidated, integrated and unifying software quality meta-framework. A typical or proposed research topic here can be formulated as illustrated below.

‘Software quality: Towards a unifying meta-framework or model’

5.6.5 The need to investigate current adoption trends of Extreme programming (XP)

Several studies have reported that XP is the main pillar and most popular or representative agile methodology (Zhang et al., 2010; Ahmed et al., 2010; Kendall, 2012). Kapitsaki and Christou (2014) contend that Scrum is fast gaining ground, when compared to other methodologies, especially XP. Meyer (2014) asserted that the assertive nature of XP compelling the use of techniques as obligations rather than as options, has comparatively hampered its overall adoption. This study corroborated the just stated two findings, by showing a decreasing adoption trend for XP. Further research in this regard is therefore recommended.

5.6.6 The need to investigate built-in quality abilities in agile methodologies.

Previous studies have proclaimed that agile methodologies have built-in quality abilities that lead to improved software quality (Huo et al., 2004; Bhasin, 2012). This study found that 60% of agile users are still employing software quality standards in their system development processes, while 40% are not. Why would more than half of agile users still use other quality standards if the built-in quality enhancing ability assertion is correct? This raises the question as to whether the software quality was improved as a consequence of using pure agile methods (i.e. without any external quality standard), or whether as a result of software quality standards used. A typical or proposed research topic here might be formulated as follows:

‘An empirical validation of built in quality abilities in agile methodologies’.

5.6.7 The need to investigate claimed methodology use versus actual use.

It was previously noted that there is a fundamental disconnect between theory and practice that requires closure (see section 5.2.22 and 4.4). Users often claim to be using a particular methodology, but there is never adequate guarantee or assurance that they are actually adhering to the principles or philosophy of the methodology in its entirety. For example, users might claim to be completely (100%) using methodology X, meanwhile actual work being done using methodology X is 15 percent. This will lead to wrong and deceitful conclusions if software quality is improved or compromised. An investigation into claimed methodology use versus actual use can help uncover insights where the misalignment between theory and practice lies.

5.6.8 The need to investigate the suitability and adaptability of agile in large and complex environments.

Several studies have reported that agile methods work well in small environments or projects (Dyba and Dingsoyr, 2008a; Barlow et al., 2011; Saxena and Kaushik, 2013), but empirical evidence of their success in general, as well as in large environments, remains scarce (Abrahamsson et al., 2010; Barlow et al., 2011). This study has corroborated those findings (see section 4.2.6). Further investigation of their suitability and adaptability in large and complex environments is therefore recommended.

5.6.9 The need for the disclosure of software quality criteria used in studies conducted.

The researcher noticed that some studies on agile methodologies and quality issues did not explicitly state which software quality criteria were under investigation (Ahmed et al, 2010). This makes comparison of studies difficult. Researchers ought to refrain from drawing broad all-encompassing conclusions about improved software quality. Software quality is a complex, multifaceted term and needs to be dissected for analysis. Research variables need to be properly defined, and operationalised, for any meaningful studies to be done and conclusions to be drawn.

5.6.10 The need for more global studies on agile software development in relation to quality.

The literature is sparse when it comes to studies investigating agile methodologies in relation to quality on a larger, wider and global scale. Very few studies have been carried on a large scale (Barlow et al., 2011) and no global study was found at the time of writing this recommendation. This lack of global studies was one of the motivations for this study. The researcher therefore recommends studies on a much larger or broader scale, so that concrete generalisations can be made, or inapplicability of agile methodologies in certain areas can become apparent.

5.6.11 The need to investigate the suitability and applicability of agile methodologies for IT offshoring.

One key feature of agile methodologies is that customer is on-site and considered as a team member, i.e. customer involvement is active and proactive. This therefore assumes that the client and the project team are co-located. This kind of assumption and model will clearly prove problematic when it comes to IT offshoring, which involves soliciting or outsourcing IT

services from a vendor in a geographically different environment. Further research on the suitability and applicability of agile methodologies in this area are needed.

5.6.12 The need to investigate the adaptability, suitability and applicability of agile methodologies in designing reusable artefacts.

One key characteristic of agile methodologies is to design only for what is presently essential. Another feature is light and minimal documentation. These assumptions will clearly prove inapplicable or inadequate for the development of reusable artefacts. Reusable artefacts include code, analysis/design documents and patterns that can be re-used from one initiative to another, either partially or in their entirety. The design of reusable artefacts involves taking a holistic and future view of the solution being designed and not only narrowly focusing on the present situation or problem. It is clearly evident that the two agile assumptions just mentioned above cannot be valid in developing reusable artefacts. Research to investigate the applicability of agile methodologies in the design of reusable artefacts is therefore recommended.

5.7 Concluding remarks

This chapter presented a synopsis and synthesis of the whole report with the objective of summarising key findings. The key findings that the research uncovered (see section 5.2 and table 29) demonstrated that the research question was answered and objectives achieved. Furthermore, the findings implicitly uncovered that theoretical assumptions about software quality in relation methodology use compared to practical reality or application are considerably different. Limitations of this study were acknowledged and the main contributions highlighted. Building on flaws in existing literature and primarily from the findings contained herein, recommendations for future research areas were proposed.

5.8 Personal reflection

This final section is a reflection on the entire dissertation and its findings.

Since the promulgation and articulation of the Agile Manifesto about a decade and a half ago, efforts to promote the agile agenda have been relentless, but largely driven by consultants and market research organisations. The amount of scientific publications on agile methodologies since then has also introduced an interesting and illuminating debate and contribution to the software engineering discipline in general. To date, there is still a great deal of competition and friction among agile and traditional methodology advocates in terms of suitability to develop software systems, with the former claiming that their paradigm is superior than that of its

predecessor when it comes to software quality improvements. No clear winner has so far emerged.

The objective of this study was emphasised throughout this report, and should be apparent to the reader by now. This dissertation would be incomplete without acknowledging the fact that some the results of this study are perfectly consistent with the researcher's initial assumptions and expectations, but that most of the findings contradicted expectations, assumptions and prior knowledge. In other words, the null hypothesis was upheld in most cases, contrary to the researcher's beliefs.

With regards to the main lessons inferred, it became apparent that, albeit possible, it is not straightforward as it seems to attribute a system quality aspect to the use of a particular methodology. The reason for this is twofold; Firstly because of the claimed methodology use versus actual use phenomenon and secondly because of the difficulty to completely control and hold elusive and clandestine extraneous variables constant in the software development environment.

Furthermore, it also became apparent that some of the researcher's initial assumptions and prior knowledge about software quality and methodology use is considerably inconsistent with the practical reality. On this note, the researcher's recommendation and advice to researchers and academics is to refrain from postulating frameworks, models, methodologies or theories from the comfort of an armchair, or from the outcome of some boardroom or resort meeting. All endeavours or efforts must be undertaken to go to the field in an effort to compare assumptions or norms, to that which actually is the case empirically, using the structured process of research.

Another point of reflexivity is the source of publication. Interpretation of research publications in light of the source is something worthy of caution. The importance of scrutinising the source of publications, whether they come from academics, consultants, governments, or from tools vendors, cannot not be over-emphasised. Publications not coming from academics should be interpreted with extreme caution.

With regards to further research concerning methodologies in general, as well as software quality in particular, the researcher's recommendation is that the possibility of combining all agile methodologies into one consolidated, systemic meta-methodology or framework should be investigated, as well as for all traditional methodologies.

Furthermore, the possibility of combining suitable and applicable existing software quality models into one systemic meta-framework or model should also be investigated. Such meta-models,

methodologies or frameworks should be vigorously and meticulously validated empirically prior to adoption. In addition, these models, frameworks or methodologies should also be continuously reviewed for their current relevance, applicability and improvements, continuously made for the benefit of software quality. These are the researcher's predictions in this regard for the not too-distant future.

From section 4.3.1, it should be recalled that the main reason why Fisher's exact test was used over Chi square test of independence was due to the fact that the contingency tables were violating the Chi square test assumption of count per cell (i.e. the expected cell frequency must not be less than 5). However, the researcher noticed that in all but one case where significant values where generated by Fisher's test, Chi square values were also significant. This observation is summarised in Table 30.

#	Variable	Correlation with:	Yes/No	*P* values (Fisher's exact test)	*P* values (Chi square values)
1.	Ease of system testing.	Traditional methodology use.	Yes	0.014	0.017
2.	Ease of system navigation.	Workshop attendance and training.	Yes	0.031	0.014
3.	Ease of system interactivity.	Active stakeholder participation on projects.	Yes	0.047	0.030
4.	Timeliness.	Traditional methodology use.	Yes	0.02	0.014
5.	Ease of system testing.	Software quality standard used.	Yes	0.017	**0.06(close)**

Table 30 : Comparison of Fisher and Chi square significant values.

The difference between Fisher and Chi square test in terms of returning significant values is almost negligible or very close; therefore, the assumption of count per cell must be reviewed and investigated by statistics students.

Agile methodologies certainly do have their place in the software engineering community, and have made an enormous and indispensable contribution. This is clearly illustrated by the proliferation of books and journals been published, ubiquitous agile conferences and the emergence of pro agile consultants.

However, they have also been a large number of scientific publications aimed at adapting them to be used in different environments. This suggests that their suitability and applicability in all environments or situations is limited. Despite this limitation, agile methods have come to stay,

and their somewhat controversial nature and debate with traditional methodologies will endure into the future.

6. REFERENCES

Abbas, N., Gravell, A. M., & Wills, G. B. (2010, August). The Impact of Organization, Project and Governance Variables on Software Quality and Project Success. In *Agile Conference (AGILE), 2010* (pp. 77-86). IEEE.

Abrahamsson, P., Conboy, K., & Wang, X. (2009). 'Lots done, more to do': the current state of agile systems development research. *European Journal of Information Systems* 18(4): 281-284.

Abrahamsson, P., Oza, N., & Siponen, M. T. (2010). Agile Software Development Methods: A Comparative Review1. In *Agile Software Development* (pp. 31-59). Springer Berlin Heidelberg.

Agile Business Analyst (online). Available from:http://www.linkedin.com/grps/Agile-Business-Analyst-1916699/about? [Accessed October 2014]

Agresti, A. (1996). *An introduction to categorical data analysis* (Vol. 135). New York: Wiley.

Ahmed, A., Ahmad, S., Ehsan, N., Mirza, E., & Sarwar, S. Z. (2010). Agile software development: Impact on productivity and quality. In *Management of Innovation and Technology (ICMIT), 2010 IEEE International Conference*: 287-291.

Aitken, A., & Ilango, V. (2013). A comparative analysis of traditional software engineering and agile software development. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on* (pp. 4751-4760). IEEE.

Ambler, S. W. (2008). Scaling Scrum: Meeting Real-World Development Needs, *Dr. Dobb's Journal* 33(5): 52-54.

Amir, M., Khan, K., Khan, A., & Khan, M. N. A. (2013). An Appraisal of agile Software Development Process. *International Journal of Advanced Science and Technology* 58: 20.

Asnawi, A. L., Gravell, A. M., & Wills, G. B. (2012, February). Emergence of agile methods: perceptions from software practitioners in Malaysia. In *AGILE India (AGILE INDIA), 2012* (pp. 30-39). IEEE.

Austin, R. D., & Devin, L. (2009). Research Commentary-Weighing the Benefits and Costs of Flexibility in Making Software: Toward a Contingency Theory of the Determinants of Development Process Design. *Information Systems Research*, *20*(3), 462-477.

Avison, D., & Fitzgerald, G. (2006). *Information systems development: methodologies, techniques and tools*. 4th edition, London: McGraw Hill.

Barlow, J., Giboney, J., Keith, M., Wilson, D., Schuetzler, R., Lowry, P., & Vance, A. (2011). Overview and guidance on agile development in large organizations. *Communications of the Association for Information Systems* 29(2): 25-44.

Beck, K. (2000). *Extreme programming explained: Embrace change*, Boston: Addison-Wesley Professional.

Beck, K. (2003). *Test-driven development: By example*, Boston: Addison-Wesley Professional.
Bedell, C. 2011. Agile Methodology adoption in the decline (online). Available from: http://searchsoftwarequality.techtarget.com/feature/Agile-methodology-adoption-in-decline [Accessed August2014]

Bhasin, S. (2012, February). Quality assurance in agile: a study towards achieving excellence. In *AGILE India (AGILE INDIA), 2012* (pp. 64-67). IEEE.

Boehm, B. (2002). Get ready for agile methods, with care. *Computer*, 35(1): 64-69.

Boehm, B., & Turner, R. (2003). Using risk to balance agile and plan-driven methods. *Computer* 36(6): 57-66.

Boehm, B., & Turner, R. (2004). Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods. In *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on* (pp. 718-719). IEEE.

Boehm, B., & Turner, R. (2005). Management challenges to implementing agile processes in traditional development organizations. *Software, IEEE* 22(5): 30-39.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, *21*(5), 61-72.

Boehm B.W, Brown J.R, and Lipow M (1976), "Quantitative Evaluation of Software Quality," International Conference on Software Engineering, *Proceedings of the 2nd international conference on Software engineering.*

Bonner, N. A., Teng, J. T., & Nerur, S. P. (2010). The Perceived Advantage of Agile Development Methodologies by Software Professionals: Testing an Innovation-Theoretic Model. In *AMCIS* (p. 93).

Breivold, H. P., Sundmark, D., Wallin, P., & Larsson, S. (2010, August). What does research say about agile and architecture? In *Software Engineering Advances (ICSEA), 2010 Fifth International Conference on* (pp. 32-37). IEEE.

Brooks, F. P. (1987). No Silver Bullet. Essence and Accidents of Software Engineering. *Computer* 20(4): 10-19.

Cao, L., Mohan, K., Xu, P., & Ramesh, B. (2009). A framework for adapting agile development methodologies. *European Journal of Information Systems* 18(4): 332-343.

Charvat, J. (2003). *Project management methodologies: Selecting, implementing, and supporting methodologies and processes for projects*, London: John Wiley and Sons.

Cockburn, A. (2004). *Crystal clear: A human-powered methodology for small teams*, London: Pearson Education.

Cockburn, A., & Highsmith, J. (2001). Agile software development: The people factor. *Computer* 34(11): 131-133.

Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research* 20(3): 329-354.

Conboy, K., & Fitzgerald, B. (2004). Toward a conceptual framework of agile methods. In *Extreme Programming and Agile Methods-XP/Agile Universe 2004* (pp. 105-116). Springer Berlin Heidelberg.

Cook, T. D., & Reichardt, C. S. (Eds.). (1979). *Qualitative and quantitative methods in evaluation research.* Beverly Hills, CA: Sage.

Cooper, D. R., & Schindler, P. S. (2006). Business research methods: Empirical investigation. *Journal of Service Research* 1(2): 108-28.

Coronel, C., Morris, S., & Rob, P. (2009). *Database systems: Design, implementation, and management*. Cengage Learning.

Create Surveys, get answers (online) Available from: https://www.surveymonkey.com/ [Accessed September 2015]

Chrissis, M.B., Konrad, M. and Shrum, S., (2003). *CMMI guidelines for process integration and product improvement*. Addison-Wesley Longman Publishing Co., Inc.

Creswell, J. W. (2003). Research design. *Qualitative, quantitative and mixed methods approaches*.

Cunningham, W. 2001.Manifesto for Agile Software development (online). Available from: http://agilemanifesto.org [accessed June 2014]

Cunningham, W. 2001.Principles behind the Agile Manifesto (online). Available from: http://www.agilemanifesto.org/principles.html [Accessed September 2013]

Daft, R. L. (1983). Learning the craft of organizational research. *Academy of Management Review,* 8: 539-546.

Delone, W. H., & McLean, E. R. (2003). The DeLone and McLean Model of Information Systems Success: A Ten-Year Update. *Journal of Management Information Systems* 19(4): 9-30.

DeLone, W. H., & McLean, E. R. (1992). Information systems success: the quest for the dependent variable. *Information systems research* 3(1): 60-95.

Dey, I. (1993). Qualitative Data Analysis. London: Routledge.

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software* 85(6): 1213-1221.

Dromey R.G (1995) "A Model for Software Product Quality", *IEEE Transactions on Software Engineering*, no. 2, pp. 146-163

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and software technology* 50(9):833-859.

Dybå, T., & Dingsøyr, T. (2008, October). Strength of evidence in systematic reviews in software engineering. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement* (pp. 178-187). ACM.

Dyba, T., & Dingsoyr, T. (2009). What do we know about agile software development? *Software* 26(5): 6-9.

Erickson, J., Lyytinen, K., & Siau, K. (2005). Agile modeling, agile software development, and extreme programming: the state of research. *Journal of Database Management* 16(4):88-100.

Esfahani, H. C., Yu, E., & Annosi, M. C. (2010). Capitalizing on empirical evidence during agile adoption. *In agile Conference AGILE,* IEEE: 21-24.

Everitt, B. S. (1992). *The analysis of contingency tables*, New York: CRC Press.

Farrokh, J. and Mansur, A.K., (2013). Project Management Maturity Models and Organizational Project Management Maturity Model (OPM3): A Critical Morphological Evaluation. In *Proceedings of World Academy of Science, Engineering and Technology* (No. 77, p. 48). World Academy of Science, Engineering and Technology (WASET).

Felsing, J. M., & Palmer, S. R. (2002). A Practical Guide to Feature-Driven Development. *IEEE Software* 7: 67-72.

Ferenc, R., Hegedűs, P. and Gyimóthy, T., (2014). Software product quality models. In *Evolving software systems* (pp. 65-100). Springer Berlin Heidelberg.

Fraenkel, J. R., & Wallen, N. (1993). *How to design and evaluate research in education.* New York, NY: McGraw-Hill.

Gable, G. G., Sedera, D., & Chan, T. (2008). Re-conceptualizing information system success: The IS-impact measurement model. *Journal of the association for information systems*, *9*(7), 18.

Gall, M. D., Gall, J. P., & Borg, W. R. (2007). *Educational research. Boston*, MA: Pearson Education.

Gay, L. R., Mills, G. E., & Airasian, P. (2009). *Educational research: Competencies for Analysis and applications*. Upper Saddle River, NT: Pearson Education.

Gilb, T. (1985). Evolutionary Delivery versus the "waterfall model". ACM SIGSOFT Software Engineering Notes 10(3): 49-61.

Gladden, G. R. (1982). Stop the life-cycle, I want to get off. *ACM SIGSOFT Software Engineering Notes*, *7*(2), 35-39.

Godbole, N. S. (2004). *Software quality assurance: Principles and practice*. Oxford, UK: Alpha Science International.

Healey, M.J. & Rawlinson, M.B. (1994) 'Interviewing techniques in business and management research', in V.J. Wass, V.J. and P.E. Wells (eds.). *Principles and Practice in Business Management and Research.* Aldershot: Dartmouth: 123–46.

Highsmith J (2002) *agile Software Development Ecosystem*s. Boston: Addison-Wesley.

Highsmith, J. (2004), *agile Project Management*. Boston: Addison–Wesley.

Hummel, M. (2014). State-of-the-Art: A Systematic Literature Review on agile Information Systems Development. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on* IEEE: 4712-4721.

Huo, M., Verner, J., Zhu, L., & Babar, M. A. (2004, September). Software quality and agile methods. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International* (pp. 520-525). IEEE.

IS Business Analyst (online) Available from: http://www.linkedin.com/grps/IS-Business-Analyst-46944/about? [Accessed October 2014].

Jayaratna, N. (1994). *Understanding and Evaluating Methodologies: NIMSAD, a Systematic Framework*. Maidenhead, UK: McGraw-Hill.

Jiang, L., & Eberlein, A. (2009,). An analysis of the history of classical software development and agile development. In *Systems, Man and Cybernetics, SMC 2009. IEEE International Conference on*: 3733-3738.

Johnson, R. R. B., and Christensen, L. B. (2010). *Educational research: Quantitative, qualitative, and mixed approaches*. Sage Publications.

Jokela, T., & Abrahamsson, P. (2004). Usability assessment of an extreme programming project: Close co-operation with the customer does not equal to good usability. In *Product focused software process improvement* (pp. 393-407). Springer Berlin Heidelberg.

Juran, J. M. (1992). *Juran on quality by design: The new steps for planning quality into goods and services*. New York: Free Press.

Schwaber, K. & Beedle, M. (2001). *Agile Software Development with Scrum.* Upper Saddle River: Prentice Hall.

SPSS Statistics (online) Available from: http://www-01.ibm.com/software/analytics/spss/products/statistics/ [Accessed October 2014]

Kapitsaki, G. M., & Christou, M. (2014). Where is Scrum in the current agile world? In *Evaluation of Novel Approaches to Software Engineering (ENASE), 2014 International Conference on* IEEE. (pp. 1-8).

Kendall, K. E., Kong, S., & Kendall, J. E. (2010). The Impact of agile Methodologies on the Quality of Information Systems: Factors Shaping Strategic Adoption of agile Practices. *International Journal of Strategic Decision Sciences,* 1(1): 41-56.

Khan, A. I., Qurashi, R. J. & Usman A. K. (2011). A comprehensive study of commonly practiced heavy and light weight software methodologies. *International Journal of Computer Science Issues* 8(4):441-450.

Knox, K. (2004). A Researcher's Dilemma: Philosophical and Methodological Pluralism. *Electronic Journal of Business Research Method,* 2(2): 119-128.

Kumar, G., & Bhatia, P. K. (2014). Comparative Analysis of Software Engineering Models from traditional to Modern Methodologies. In *Advanced Computing and Communication Technologies (ACCT), 2014 Fourth International Conference on* (pp. 189-196). IEEE.

Laanti, M., Salo, O. & Abrahamsson, P. (2011). Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. *Information and Software Technology* 53(3): 276-290.

Larman, C. (2004). *Agile and iterative development: a manager's guide*. Addison-Wesley Professional.

Larman, C., and Basili, V. R. (2003). Iterative and incremental development: A brief history. *Computer*, 36(6): 47-56.

Leau, Y. B., Loo, W. K., Tham, W. Y. and Tan, S. F. (2012). Software Development Life Cycle AGILE vs. traditional Approaches. In *International Conference on Information and Network Technology* 37(1): 162-167.

Lee, J. C. (2006). Embracing agile development of usable software systems. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems,* ACM: 1767-1770.

Leffingwell, D. (2007). *Scaling software agility: Best practices for large enterprises*. Addison-Wesley Professional.

McCall J.A, Richards P.K, and Walters G.F (1977),"Factors in Software Quality", *Nat'l Tech. Information Service,* no. Vol. 1, 2 and 3.

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision support systems* 15(4): 251-266.

McDonald, J. H. (2009). *Handbook of biological statistics Vol. 2*. Baltimore, MD: Sparky House Publishing: 173-181.

McInerney, P., & Maurer, F. (2005). UCD in agile projects: Dream team or odd couple? *Interactions* 12(6): 19-23.

McKinney, V., Kanghyun, Y., & Zahedi, F. M. (2002). The measurement of web-customer

Satisfaction: An expectation and disconfirmation approach. *Information Systems Research* 13(3): 296–315.

Merisalo-Rantanen, H., Tuure, T. & Matti R. (2005). Is extreme programming just old wine in new bottles: A comparison of two cases, *Journal of Database Management* 16 (4):41–61.

Merriam, S.B. (2002). *Qualitative research and cased study applications in education.* San Francisco: Jossey-Bass Publishers.

Meso P, R. Jain (2006). Agile software development: Adaptive systems principles and best practices, *Information Systems Management* 23(3): 19–30.

Meyer, B. (2014). Agile methods. In *agile!* Springer International Publishing: 133-143.

Miller, S. I., & Fredericks, M. (1988). Uses of metaphor: A qualitative case study. *International Journal of Qualitative Studies in Education*, *1*(3), 263-272.

Mitchell, V. (1996). Assessing the reliability and validity of questionnaires: an empirical example. *Journal of Applied Management Studies*, *5*, 199-208.

Mnkandla, E., & Dwolatzky, B. (2006, October). Defining agile software quality assurance. In *Software Engineering Advances, International Conference on* (pp. 36-36). IEEE.

Mohammad, A. H., & Alwada'n, T. (2013). Agile Software Methodologies: Strength and Weakness. *International Journal of Engineering Science and Technology* 5 (3):455-459.

Moniruzzaman, A. B. M., & Akhter Hossain, S. (2013). Comparative Study on agile software development methodologies. *ArXiv preprint arXiv: 1307.3356.*

Mordal, K., Anquetil, N., Laval, J., Serebrenik, A., Vasilescu, B. and Ducasse, S., (2013). Software quality metrics aggregation in industry. *Journal of Software: Evolution and Process*, *25*(10), pp.1117-1135.

Mouton, J. (2005) *How to succeed in your Masters and Doctoral Studies*. 9th edition. Pretoria: Van Schaik.

Nardi, P. M. (2002). *Doing Survey Research: A Guide to Quantitative Research Methods*. Allyn and Bacon.

Nerur, S., Mahapatra, R., and Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM* 48(5): 72-78.

Newman, I., & Benz, C. R. (1998). *Qualitative–quantitative research methodology: Exploring the interactive continuum.* Carbondale, IL: Southern Illinois University Press.

Nikiforova, O., Nikulsins, V, & Sukovskis, U (2009): Integration of MDA Framework into the Model of traditional Software Development. *Frontiers in Artificial Intelligence and Applications, Databases and Information Systems* 187:229–239.

Oates, BJ. 2006. *Researching information systems and computing*. London: Sage.

Olivier, M. S. (2009). *Information technology research: a practical guide for computer science and informatics*. Pretoria: Van Schaik.

Onwuegbuzie, A. J., & Leech, N. L. (2005). On becoming a pragmatic researcher: The importance of combining quantitative and qualitative research methodologies. *International Journal of Social Research Methodology* 8(5), 375-387.

Pathak, K., & Saha, A. (2013). Review of agile Software Development Methodologies. *International Journal*, *3*(2).

Perry, D. E., Porter, A. A., & Votta, L. G. (2000). Empirical studies of software engineering: a roadmap. In *Proceedings of the conference on the future of Software engineering*. ACM: 345-355.

Phillips, D. C., & Burbules, N. C. (2000). *Postpositivism and educational research.* Lanham, MD: Rowman and Littlefield.

Poppendieck, M., & Poppendieck, T. (2003). *Lean software development: an agile toolkit*. Addison-Wesley Professional.

Prause, C. R., & Durdik, Z. (2012). Architectural design and documentation: Waste in agile development? In *Software and System Process (ICSSP), 2012 International Conference on* IEEE: 130-134.

Project Management Office (online) Available from:http://www.linkedin.com/grps/PMO-Project-Management-Office-80342/about? [Accessed October 2014].

Rao K. N., Naidu G. K., & Chakka.P (2011) A study of the agile software development methods, applicability and implications in industry, *International Journal of Software Engineering and Its Applications* 5: 5-45.

Robson, C. (2002) *Real World Research* (2nd ed.). Oxford: Blackwell.

Rodriguez Gonzalez, P., Yagüe Panadero, A., Alarcón Cavero, P. P., & Garbajosa Sopeña, J. (2009). Some Findings Concerning Requirements in agile Methodologies. *Product-Focused Software Process Improvement* 32(4):171-184.

Pressman, R. S. (2010). Software engineering: a practitioner's approach, 7th edition. New York: McGraw-Hill.

Royce, W. W. (1970). Managing the development of large software systems. *Proceedings of IEEE WESCON.* 1-9.

Salkind, N. J. (Ed.). (2010). *Encyclopedia of research design* (Vol. 1). Sage.

Salo, O., & Abrahamsson, P. (2007). An iterative improvement process for agile software development. *Software Process: Improvement and Practice*, 12(1): 81-100.

Salo, O., & Abrahamsson, P. (2008). Agile methods in European embedded software development organisations: A survey on the actual use and usefulness of Extreme Programming and Scrum. *Software, IET*, 2(1): 58-64.

Santos, M.A., Bermejo, P. H. S., Oliveira, MS &Tonelli, A. O. (2011). Agile Practices: An Assessment of Perception of Value of Professionals on the Quality Criteria in Performance of Projects". *Journal of Software Engineering and Applications* 4: 700-709.

Saunders, M. Lewis, P. &Thornhill, A. 2009. *Research methods for business students*. 5th edition. Harlow: Prentice Hall.

Saxena, P., & Kaushik, M. (2013). Software Development: Techniques and Methodologies. *International Journal of Software and Hardware Research in Engineering* 1(3):48-52.

Schmidt, C. T., Ganesha Venkatesha, S., & Heymann, J. (2014). Empirical insights into the perceived benefits of agile software engineering practices: a case study from SAP. In *Companion Proceedings of the 36th International Conference on Software Engineering,* ACM: 84-92.

Schwaber, C., Leganza, G., & D'Silva, D., 2007. The Truth about agile Processes, Forrester Research.

Sedera, D., & Gable, G. (2004). A factor and structural equation analysis of the enterprise systems success measurement model. *ICIS 2004 Proceedings*, 36.

Selic, B. (2009) "agile Documentation, Anyone?", *IEEE Software* (26)6: 11–12.

Sfetsos, P., & Stamelos, I. (2010). Empirical studies on quality in agile practices: A systematic literature review. In *Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the* IEEE: 44-53.

Sherehiy, B., Karwowski, W., & Layer, J. K. (2007). A review of enterprise agility: Concepts, frameworks, and attributes. International Journal of Industrial Ergonomics 37(5): 445-460.

Siau, K. & Rossi, M. (1998), Evaluation of Information Modelling Methods: A Review, *IEEE Proceedings of the 31st Annual Hawaii International Conference on Systems Sciences*: 314 –322.

Sieber, S. D. (1973). The integration of fieldwork and survey methods. *American Journal of Sociology* 73: 1335–1359.

Singh, B. and Kannojia, S.P., 2013, April. A review on software quality models. In *Communication Systems and Network Technologies (CSNT), 2013 International Conference on* (pp. 801-806). IEEE.

Sjoberg, D. I., Dyba, T., and Jorgensen, M. (2007). The future of empirical methods in software engineering research. In *Future of Software Engineering, 2007.* IEEE: 358-378.

Sommerville, I. (2007) *Software engineering,* 8th edition. Boston, MA: Addison-Wesley.

Stapleton, J. (Ed.). (2003). *DSDM: Business focused development*. New York: Pearson Education.

Schach, S. (2010). *Object-Oriented and Classical Software Engineering*. 8th Edition. New York: McGraw-Hill.

Sy, D. (2007). Adapting usability investigations for agile user-centered design. *Journal of Usability Studies* 2(3): 112-132.

Szalvay, Victor (2004). *An introduction to agile software development.* Danube Technologies: 1-9.

Twidale, M. B., and Nichols, D. M. (2013). Agile Methods for agile Universities. In *Re-imagining the Creative University for the 21st Century* (pp. 27-48). Sense Publishers.

University of South Africa (2007). *Policy on Research Ethics*. Available from: Available from: http://www.unisa.ac.za/contents/research/docs/researchethicspolicy_apprvcounc_21sept07.pdf (accessed September 2014).

Urbach, N., and Müller, B. (2012). The updated DeLone and McLean model of information systems success. In *Information Systems Theory.* New York, Springer: 1-18.

Van Koten, C. and Gray, A.R., (2006). An application of Bayesian network for predicting object-oriented software maintainability. *Information and Software Technology*, *48*(1), pp.59-67.

Valerdi, R., and Davidz, H. L. (2009). Empirical research in systems engineering: challenges and opportunities of a new frontier. *Systems Engineering* 12(2): 169-181.

Vinekar, V., Slinkman, C. W., and Nerur, S. (2006). Can agile and traditional systems development approaches coexist? An ambidextrous view. *Information systems management*: 23(3): 31-42.

Wagner, S., Lochmann, K., Heinemann, L., Kläs, M., Trendowicz, A., Plösch, R., Seidl, A., Goeb, A. and Streit, J., (2012). The quamoco product quality modelling and assessment approach. In *Proceedings of the 34th international conference on software engineering* (pp. 1133-1142). IEEE Press.

Weinberg, G. M. (1992). *Quality software management (Vol. 1): Systems thinking*. Dorset: House Publishing.

Whitten, J. & Bentley, L. D. (2007). *Systems analysis and design methods.* 7th Edition. New York: McGraw-Hill.

Wysocki, R. K. (2011). *Effective project management: traditional, agile, extreme*, Indiana: John Wiley & Sons.

Zhang, X., Hu, T., Dai, H., & Li, X. (2010). Software development methodologies, trends, and implications. *Information Technology Journal*, *9*(8).

Zhou, Y. and Xu, B., (2008). Predicting the maintainability of open source software using design metrics. *Wuhan University Journal of Natural Sciences*, *13*(1), pp.14-20.

7. APPENDIX: QUESTIONNAIRE

Agile and conventional methodologies : An empirical investigation of their impact on software quality parameters

SECTION A : BACKGROUND INFORMATION.

1. Which software development methodology do you use in your organisation or projects?(Examples of traditional methodologies are waterfall or SDLC,Spiral model etc and examples of agile methdologies are Scrum,XP etc)

- ○ Traditional software development methodologies
- ○ Agile software development methodologies
- ○ Both methodologies
- ○ Other (please specify)

2. What does the term Agile methodology mean to you? Select all that apply

- ☐ Individuals and Interactions over Processes and Tools
- ☐ Working software over Comprehensive Documentation
- ☐ Customer Collaboration over Contract negotiation
- ☐ Responding to Change over following a Plan
- ☐ Iterative and Incremental development
- ☐ All of the above
- ☐ Other (please specify)

3. Which Agile methodology does your organisation or project practice?

- ○ Lean development
- ○ Crystal methodologies
- ○ Dynamic system development method
- ○ Feature driven development
- ○ Extreme programming(XP)
- ○ Scrum
- ○ Kanban
- ○ Adaptive system development
- ○ Not sure
- ○ Other or a combination of agile methodologies (please specify)

4. How long has your organization or project been practicing Agile?

- ◯ Less than a year
- ◯ 1-2 years
- ◯ 3-5 years
- ◯ Greater than 6 years

5. What is the average experience of your agile team?

- ◯ Less than a year
- ◯ 1-2 years
- ◯ 3-5 years
- ◯ Greater than 6 years

6. What is the total number of people in your project team?

- ◯ 1-10
- ◯ 11-100
- ◯ Greater than 100

7. What was the average size of the projects that you worked on in terms of cost and complexity?

- ◯ Small(Less than 10 developers)
- ◯ Medium(11-20 developers)
- ◯ Large(More than 20 developers)

8. What is the total number of people in your organization?

- ◯ 1-10
- ◯ 11-100
- ◯ 101-1000
- ◯ 1001-10,000
- ◯ 10001-100,000
- ◯ Over 100,000

9. Which of the following software quality standards or frameworks do you comply with or adhere to?

- ◯ CMM
- ◯ CMMI
- ◯ PMMM
- ◯ ISO 9000
- ◯ None
- ◯ Other or a combination of frameworks (please specify)

10. Project stakeholders (Users, Management and Sponsor) actively participated in the projects that i worked on.

- ◯ Strongly agree
- ◯ Agree
- ◯ Neutral
- ◯ Disagree
- ◯ Strongly disagree

11. Your project team regularly attends mandatory relevant on the job training workshops or courses?

- ◯ Strongly disagree
- ◯ Agree
- ◯ Neutral
- ◯ Disagree
- ◯ Strongly agree

12. Which of the following best describes your current position within your organization?

- ◯ Business stakeholder
- ◯ Developer/programmer
- ◯ Project manager
- ◯ Programme/Portfolio manager
- ◯ Test Analyst
- ◯ Quality assurance
- ◯ Architect
- ◯ Business/System Analyst
- ◯ Consultant
- ◯ Other (please specify)

13. Which of the following best describes the identity of your organization?

- ○ Banking/insurance
- ○ Computer related (Hardware/Software)
- ○ Telecommunications
- ○ Academic/Research
- ○ Real estate
- ○ Business supplies/services
- ○ Entertainment/media/publishing
- ○ Petroleum
- ○ Hospitality
- ○ Medical/health care
- ○ Government
- ○ Engineering/construction
- ○ Consulting
- ○ Legal services
- ○ Mining
- ○ Manufacturing/distribution
- ○ Consumer retail/wholesale
- ○ Non-profit/membership organization
- ○ Electrical machines
- ○ Aerospace
- ○ Other (please specify)

14. In what country do you currently reside?

SECTION B.REASERCH VARIABLES: SYSTEM AND INFORMATION QUALITY

Please kindly use the information provided in the tables to complete the questions following them.

15.

System parameter: EASE OF SYSTEM MAINTENANCE

Systems	System X	System Y
Project characteristics : Documentation	Enormous and detailed documentation produced	Light and minimal documentation produced

Given the two systems above from the projects that you worked on, which of them is likely to be easier to maintain?

- ○ System X
- ○ System Y
- ○ Neutral

Please kindly use the information provided in the table to complete the question.

16.

System parameter: EASE OF SYSTEM TESTING

Systems	System X	System Y
Project characteristics : Documentation	Enormous and detailed documentation produced	Light and minimal documentation produced

Given the two systems above from the projects that you worked on,which of them is likely to be easier to test to uncover defects?

- ○ Neutral
- ○ System X
- ○ System Y

Please kindly use the information provided in the table to complete the question.

17.

System parameter : EASE OF SYSTEM TRAINING

Systems	System X	System Y
Project characteristics : Documentation	Enormous and detailed documentation produced	Light and minimal documentation produced

Given the two systems above from the projects that you worked on,which of them is likely to be easier to train system users?

- ○ System X
- ○ System Y
- ○ Neutral

Please kindly use the information provided in the table to complete the question.

18.

System parameter : EASE OF SYSTEM LEARNING

Systems	System X	System Y
Project characteristics : Documentation	Enormous and detailed documentation produced	Light and minimal documentation produced

Given the two systems above from the projects that you worked on,which of them is likely to be easier to learn by system users?

- ○ System X
- ○ Neutral
- ○ System Y

Please kindly use the information provided in the table to complete the question.

19.

System parameter : ROBUSTNESS OF SYSTEM ARCHITECTURE

Systems	System X	System Y
Project characteristics : System architecture	Design architecture for current and future requirements	Design architecture for only what is presently essential

Given the two systems above from the projects that you worked on,which of them is likely to withstand extensions with the addition of new components or functionality without failing?

- ○ System X
- ○ System Y
- ○ Neutral

Please kindly use the information provided in the table to complete the question.

20.

System parameter : PORTABILITY

Systems	System X	System Y
Project characteristics : System architecture	Design architecture for current and future requirements	Design architecture for only what is presently essential

Given the two systems above from the projects that you worked on,which of them is likely to be easier to install on different hardware and software platforms?

- ○ System X
- ○ Neutral
- ○ System Y

Please kindly use the information provided in the table to complete the question.

21.

System parameter : MEET USABILITY NEEDS

Systems	System X	System Y
Project characteristics : Customer involvement	**Customer involvement is low and passive**	• **Customer is onsite and considered as a team member** • **Customer involvement is active and proactive**

Given the two systems above from the projects that you worked on,which of them is likely to be easier to use?

- ○ System X
- ○ Neutral
- ○ System Y

Please kindly use the information provided in the table to complete the question.

22.

System parameter : EASE OF CUSTOMISATION

Systems	System X	System Y
Project characteristics : Customer involvement	**Customer involvement is low and passive**	• **Customer is onsite and considered as a team member** • **Customer involvement is active and proactive**

Given the two systems above from the projects that you worked on,which of them is likely to be easier to customize or adapt according to personal preferences?

- ○ System X
- ○ Neutral
- ○ System Y

Please kindly use the information provided in the table to complete the question.

23.

System parameter : EASE OF NAVIGATION

Systems	System X	System Y
Project characteristics : Customer involvement	**Customer involvement is low and passive**	• **Customer is onsite and considered as a team member** • **Customer involvement is active and proactive**

Given the two systems above from the projects that you worked on,which of them is likely to be easier to move around from one page or functionality to another?

- ◯ System X
- ◯ Neutral
- ◯ System Y

Please kindly use the information provided in the table to complete the question.

24.

System parameter : EASE OF INTERACTIVITY

Systems	System X	System Y
Project characteristics : Customer involvement	**Customer involvement is low and passive**	• **Customer is onsite and considered as a team member** • **Customer involvement is active and proactive**

Given the two systems above from the projects that you worked on,the degree of interactivity is likely to be easier in which one?

- ◯ System Y
- ◯ System X
- ◯ Neutral

Please kindly use the information provided in the table to complete the question.

25.

System parameter : ERROR MESSAGE COMPREHENSIBILITY

Systems	System X	System Y
Project characteristics : Customer involvement	**Customer involvement is low and passive**	• **Customer is onsite and considered as a team member** • **Customer involvement is active and proactive**

Given the two systems above from the projects that you worked on, the user is likely to easily understand error messages from which one?

- ◯ System Y
- ◯ System X
- ◯ Neutral

Please kindly use the information provided in the table to complete the question.

26.

System parameter : HELP FACILITIES

Systems	System X	System Y
Project characteristics : Customer involvement	**Customer involvement is low and passive**	• **Customer is onsite and considered as a team member** • **Customer involvement is active and proactive**

Given the two systems above from the projects that you worked on,which of them is likely to produce adequate help facilities that the user can easily understand?

- ◯ System Y
- ◯ System X
- ◯ Neutral

Please kindly use the information provided in the table to complete the question.

27.

System parameter : SYSTEM CORRECTNESS

Systems	System X	System Y
Project characteristics : Requirements	Requirements are predicted early, fairly stable, clearly defined and documented	Requirements are emergent, rapid change, unknown and discovered during the project

Given the two systems above from the projects that you worked on,which of them is likely to function according to a defined specification?

- ◯ System Y
- ◯ System X
- ◯ Neutral

Please kindly use the information provided in the table to complete the question.

28.

System parameter : TIMELINESS

Systems	System X	System Y
Project characteristics : Project life cycle model	Linear/sequential project life cycle model	Evolutionary, incremental and iterative project life cycle model

Given the two systems above from the projects that you worked on,which of them is likely to be delivered on time?

- ◯ Neutral
- ◯ System Y
- ◯ System X

Please kindly use the information provided in the table to complete the question.

29.

System parameter : COMPLETENESS OF SYSTEM FEATURES

Systems	System X	System Y
Project characteristics : Project life cycle model	Linear/sequential project life cycle model	Evolutionary,incremental and iterative project life cycle model

Given the two systems above from the projects that you worked on,which of them is likely to be delivered with complete features?

- ◯ System Y
- ◯ Neutral
- ◯ System X

30. Which of the following project life cycle models was most suitable for your projects?

- ○ Linear/sequential model
- ○ Evolutionary,incremental and iterative model
- ○ Synchronize and stabilize model
- ○ Spiral life cycle model
- ○ Rapid prototyping model
- ○ Code and fix model
- ○ Another life cycle model or a combination of life cycle models (please specify)